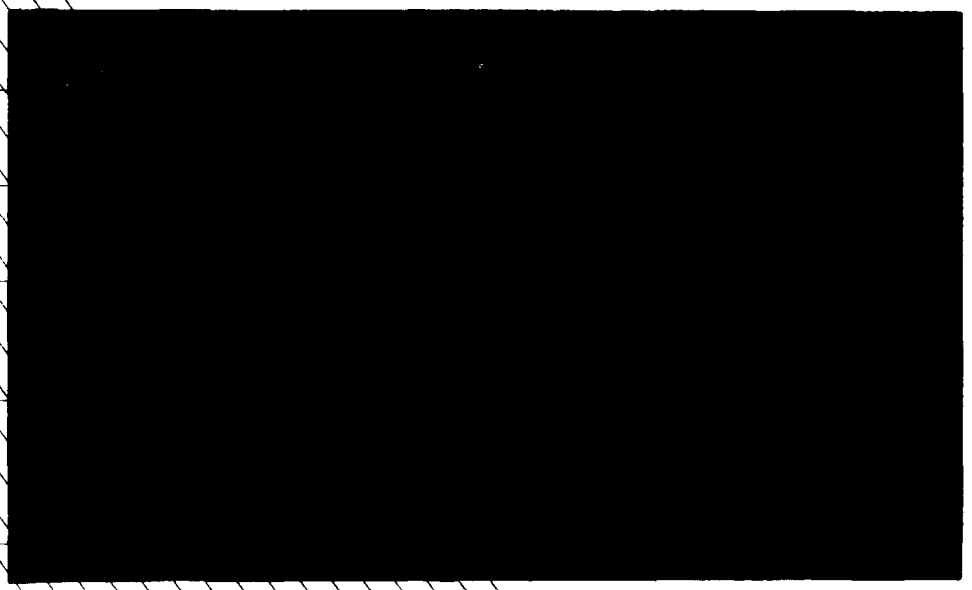
(2)

Computing About Physical Objects



Purdue University
Department of
Computer Sciences

(2)

SURFACE APPROXIMATIONS IN GEOMETRIC MODELING*

Jung-Hong Chuang

Computer Sciences Department
Purdue University
Technical Report CSD-TR-1023
CAPO Report CER-90-37
September, 1990

* A Thesis submitted to the faculty of Purdue University in partial fulfillment of the requirements for the degree of Doctor of Philosophy, August 1990.

SURFACE APPROXIMATIONS IN GEOMETRIC MODELING

A Thesis

Submitted to the Faculty

of

Purdue University

by

Jung-Hong Chuang

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

August 1990

To my parents

ACKNOWLEDGMENTS

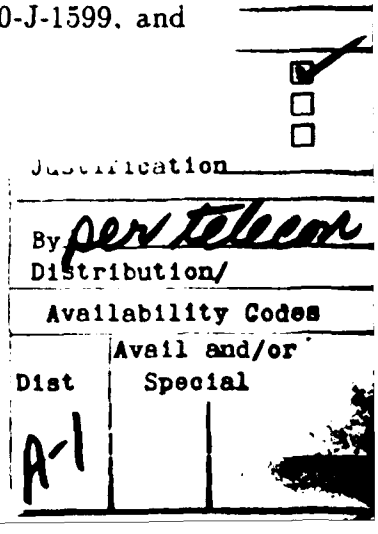
I would like to express my deepest gratitude to my advisor, Professor Christoph M. Hoffmann, for his guidance and encouragement, and for his detailed reading and valuable feedback. He has significantly influenced my professional conduct and training. Professors Robert Lynch, Bradley J. Lucier and Elias N. Houstis deserve special thanks for their interest in my research and for serving on my thesis committee. I am also thankful to Professor Chanderjit Bajaj for his interest with several aspects of my study. Thanks to Kevin Kuehl for his great effort to implement the algorithms on piecewise linear approximations. Thanks also to Pamela Vermeer for reviewing this manuscript.

I thank all the friends who have offered assistance and encouragement during my stay at Purdue. Notable among these are Danny Chen, Ching-Shoei Chiang, Shy-Renn Lian, Hsin Pan, Vic Pollara, Preeti Shrikhande, Charles and Jui-Fen Trappey, Jyh-Jong Tsay, Ko-Yang Wang, JiaXun Yu, and Jianhua Zhou.

My wife Whey-Ming Tina Song deserves special thanks for her encouragement, love, patience, and understanding. Such as they are, my accomplishments belong equally to her.

Finally, to my parents, who seemed to be proud of what I did, I would like to express my deepest appreciation. This thesis is dedicated to them.

I would like to acknowledge the financial support provided by the Office of Naval Research under contracts N00014-86-K-0465 and N000014-90-J-1599, and the National Science Foundation under grant CCR-86-19817.



Statement "A" per telecon Dr. Andre van Tilborg. Office of Naval Research/code 1133.

VHG 11/28/90

TABLE OF CONTENTS

Page

Page

LIST OF FIGURES

Figure Page

LIST OF TABLES

ABSTRACT

Chuang, Jung-Hong. Ph.D., Purdue University, August 1990. Surface Approximations in Geometric Modeling. Major Professor: Christoph M. Hoffmann.

One of the major research efforts in the field of solid modeling focuses on extending the geometric coverage of modeling systems and on incorporating complex free-form surfaces. Some major obstacles to this goal include computing and representing intersection curves of two general surfaces, and computing and rendering very complex surfaces, including offset, Voronoi, and blending surfaces. We present local and global approximation schemes that are expected to be of practical value in overcoming the above problems. For parametric curves and surfaces, we present a method for computing an implicit approximant of low degree that approximates the curves or surface locally and achieves an order of contact that can be prescribed in advance. In principle, the method is capable of exact implicitization. Several surfaces, including offsets, blends, and Voronoi surfaces can be defined as the natural projections to $\mathbf{R}^3$ of 2-surfaces in $\mathbf{R}^n$, $n > 3$. The 2-surface in $\mathbf{R}^n$ is the zero set of a system of nonlinear equations in n variables. We present algorithms that compute the normal, tangent vectors, and normal curvatures of the projected surface directly from the nonlinear system without variable elimination. Methods are presented as well that compute the explicit and parametric approximations of the projected surface locally. Finally, for a given 2-surface in $\mathbf{R}^n$, $n > 3$, an algorithm is given that computes the piecewise linear approximation of the projected surface globally with all major computations performed in 3-space.

1. INTRODUCTION

Solid modeling has received much attention throughout the academic and industrial communities for nearly three decades. Even though significant progress has been made in basic research and in the capabilities of commercially available solid modelers, many current solid modeling systems allow only severely geometric primitives. For example PADL, a solid modeler developed at the University of Rochester by Requicha and Voelcker, only allows planar, spherical, cylindrical, conical, and toroidal faces [55].

Recent solid modeling research efforts are being directed at extending the geometric coverage of solid modelers and incorporating complex free-form surfaces. Some major obstacles to this goal have been computing and representing intersection curves of two general surfaces, and computing and rendering very complex surfaces, including offset, Voronoi, and blending surfaces. In this thesis, local and global approximation schemes are presented that are expected to be of practical value in overcoming the problems of geometric coverage.

An algebraic surface in three dimensional space can be given by an implicit representation as the polynomial equation $g(x, y, z) = 0$. Some algebraic surfaces can also be given parametrically as $x = h_1(s, t)$, $y = h_2(s, t)$, $z = h_3(s, t)$, where the h_i are polynomials or ratios of polynomials. Since both representations have their own strengths and weaknesses, many geometric computations could become simpler or practical if both representations were available and their complementary strengths could be utilized. Thus, the problem of how to convert from one representation to the other is of great practical importance. However, conversions between them are not always possible. While general techniques exist for converting from parametric to implicit form, by a process called *implicitization* based

on classical elimination theory, e.g., [46, 47, 49, 58, 65], only a subset of implicit curves and surfaces have a parametric representation. Moreover, the techniques for implicitization are extremely expensive and hence their practical use is quite limited. Since exact conversions between representations might be impossible or impractical, local approximations in terms of parametric or implicit forms should be valuable. We have derived several local approximation techniques for curves and surfaces that are defined parametrically or implicitly, and expect that these techniques will be of practical interest when they are incorporated into algorithms for surface interrogation such as computing surface intersections [23, 24].

Offset, Voronoi, and blending surfaces can be constructed, with an algebraic formulation, as the natural projection to $\mathbf{R}^3$ of 2-dimensional manifolds (2-surfaces) in a higher-dimensional space. For visualization purposes, a piecewise linear approximation (PLA) of these surfaces is desirable. As a global approximation, the PLA of parametric surfaces has been extensively studied and is a widely used tool for rendering surfaces, as well as for evaluating surface intersections. Comparatively, much less attention has been paid in the literature to piecewise approximations of implicitly defined surfaces [7, 17, 57]. We present new algorithms for computing PLAs of surfaces defined implicitly by sets of equations, including offset surfaces, Voronoi surfaces, and spherical blending surfaces.

1.1 Surface Representations in Solid Modeling

2-surfaces can be represented either in parametric or in implicit form. The implicit representation is attractive for determining directly whether a point is on the surface by checking whether or not it satisfies the implicit equation. For the parametric representation, on the other hand, it is much easier to generate points on the surface. Major parametric surfaces, including Bézier surfaces, and nonuniform rational B-splines, are attractive in interactive design because the manner

in which coefficient changes alter the surface shape qualitatively can be grasped intuitively. However, this is presently not the case for implicit surfaces.

1.1.1 Parametric Surfaces

The parametric representation of a 2-surface in $\mathbf{R}^3$ is

$$\begin{aligned} x &= x(u,v) \\ y &= y(u,v) \\ z &= z(u,v) \end{aligned} \tag{1.1}$$

and is the range of a map from $\mathbf{R}^2$ to $\mathbf{R}^3$, where u and v are usually restricted to a standard domain, say to the unit square $[0,1] \times [0,1]$. These parameter limits define a bounded rectangular piece of surface, or a surface patch. The functions $x(u,v)$, $y(u,v)$, and $z(u,v)$ customarily are polynomials or ratios of polynomials in u and v. For the major parametric surfaces used in Computer Aided Geometric Design (CAGD), the polynomials are represented in a particular basis, for instance, in the Bernstein-Bézier basis. This allows us to relate the coefficients of the coordinate functions of the surface to the geometric properties and shape of the surface, and this relationship makes parametric surfaces well-suited to interactive design. Also, many useful techniques, including subdivision and local shape control, have been developed and extensively applied in surface interrogations; see, e.g., [26]. The parametric representation can be generalized to define a 2-surface in $\mathbf{R}^n$, for $n > 3$.

1.1.2 Implicit Surfaces

An algebraic surface in $\mathbf{R}^3$ is defined as the zero set of an implicit equation

$$h(x,y,z) = 0 \tag{1.2}$$

where h is a polynomial in x, y, z. As mentioned before, efficient conversion between parametric and implicit representations would be of critical importance in

geometric computations. Sederberg [61, 63] demonstrated that it is always possible to *implicitize* a parametric curve or surface using *elimination* techniques such as resultant methods [46, 47, 49, 58]. The implicitization can also be based on Gröbner Basis techniques [18, 31, 33]. Both methods are fairly expensive and hence their practical application is currently quite limited.

Parameterization or converting an implicit representation to an equivalent parametric representation is not always possible since not all implicit surfaces can be expressed as rational parametric surfaces. In fact, only implicit curves of genus zero possess a rational parameterization; see [1, 2, 3, 4, 33, 64].

1.1.3 Implicit Surfaces in High-Dimensional Space

While many surfaces can be formulated quite easily in three dimensional space, as parametric or implicit surfaces, certain surfaces including offsets, Voronoi surfaces(equidistance surfaces), and spherical blends can not. Due to the possibility of high algebraic degrees, many geometric operations on such surfaces can lead to high computational complexity and numerical instability, and may require expensive symbolic manipulations. As an alternative, Hoffmann proposed in [30, 31] a surface representation that is defined in a higher dimensional space, with more variables but simpler equations. With this representation, complex symbolic computations and numerically delicate operations can often be avoided, and hence practical implementations can be realized.

In the following, we give the formulation of offset and Voronoi surfaces in high dimensional space.

Offset Surfaces As described in [30], the *r-offset* of the surface $g(x, y, z) = 0$ is the set of points

$$\text{Offset}(g, r) = \{\ p \mid \text{d}(g, p) = r\}$$

where $\text{d}(g, p)$ is the Euclidean distance of the point p from the surface $g = 0$. The offset surface in general has two sheets in real affine space and can be defined

mathematically applying the envelope theorem as follows:

$$\begin{aligned} (x-u)^2+(y-v)^2+(z-w)^2-r^2 &= 0 \\ g(u,v,w) &= 0 \\ (x-u,y-v,z-w)\cdot \mathbf{t}_1 &= 0 \\ (x-u,y-v,z-w)\cdot \mathbf{t}_2 &= 0 \end{aligned} \tag{1.3}$$

where (u,v,w) is the footpoint, and $\mathbf{t}_1$ and $\mathbf{t}_2$ are two linearly independent tangent directions of $g=0$ to which the direction vector $(x-u,y-v,z-w)$ must be perpendicular. For example, with $\mathbf{t}_1=(g_w,0,-g_u)$ and $\mathbf{t}_2=(0,g_w,-g_v)$, we obtain

$$\begin{aligned} g_w(x-u)-g_u(z-w) &= 0 \\ g_w(y-v)-g_v(z-w) &= 0 \end{aligned}$$

Its closed-form can be obtained in principle by eliminating u,v, and w using resultant techniques [13, 46, 61] or Gröbner basis methods [18, 19]. When $g_w=0$, $(g_w,0,-g_u)$ and $(0,g_w,-g_v)$ become linearly dependent and hence there will be an extraneous factor representing spheres on $g=0\cap g_w=0$ in its closed-form representation. To eliminate this extraneous factor, we may adjoin

$$g_u(y-v)-g_v(x-u)=0$$

to system (1.3). Moreover, the presence of singularities also introduces additional extraneous factors. Thus, the formulation in (1.3) is not *faithful* in the sense that the natural projection of the solution set of the system contains points that are not on the offset surface: see [32]. The offset formulation (1.3) is a system of four equations with degrees as high as g's while its closed-form in (x,y,z)-space has a much higher degree. For example, when g is a quadric, the closed-form of its offset may have degree 8. The computation of the intersection of the offset with other surfaces is reported in [31, 30]. Below, we define the offset of an ellipsoid as an example.

Example 1.1 Consider an ellipsoid $g(x,y,z)=4x^2+y^2+z^2-4=0$ which is centered at the origin and has semiaxes 1, 2, and 2, respectively. The offset of

$g = 0$ by the distance 2 is formulated as

$$\begin{aligned}(x-u)^2+(y-v)^2+(z-w)^2-4 &= 0\\ 4u^2+v^2+w^2-4 &= 0\\ w(x-u)-4u(z-w) &= 0\\ w(y-v)-v(z-w) &= 0\end{aligned} \tag{1.4}$$

When $w = 0$, $(w, 0, -4u)$ and $(0, w, -v)$ become linearly dependent. Hence the closed-form of the offset of g in (x, y, z)-space reveals an extraneous factor z and has degree 9. ■

Voronoi Surfaces The *Voronoi surface* of two given surfaces $g(x, y, z) = 0$ and $h(x, y, z) = 0$, denoted as $\mathrm{Vor}(g, h)$, is the locus of points equidistant from g and h. It is formally defined by

$$\mathrm{Vor}(g,h) = \{\, p \in \mathbf{R}^3 \mid \mathrm{d}(g,p) = \mathrm{d}(h,p)\}$$

The Voronoi surface proves useful in defining constant-radius blends [32], variable-radius blends [21, 30], and skeletons (medial-axis surfaces) of an object [25]. As shown in [30], we define the $\mathrm{Vor}(g, h)$ as the common points of the offsets from both g and h by an identical but unspecified distance r. Thus, $\mathrm{Vor}(g, h)$ can be formulated as eight equations in ten variables

$$\begin{aligned}(x-u)^2+(y-v)^2+(z-w)^2-r^2 &= 0\\ g(u,v,w) &= 0\\ (x-u,y-v,z-w)\cdot \mathbf{t}_1 &= 0\\ (x-u,y-v,z-w)\cdot \mathbf{t}_2 &= 0\\ (x-\bar{u})^2+(y-\bar{v})^2+(z-\bar{w})^2-r^2 &= 0\\ g(\bar{u},\bar{v},\bar{w}) &= 0\\ (x-\bar{u},y-\bar{v},z-\bar{w})\cdot \bar{\mathbf{t}}_1 &= 0\\ (x-\bar{u},y-\bar{v},z-\bar{w})\cdot \bar{\mathbf{t}}_2 &= 0\end{aligned} \tag{1.5}$$

where (u, v, w) and $(\bar{u}, \bar{v}, \bar{w})$ are footpoints on g and h respectively, and $(\mathbf{t}_1, \mathbf{t}_2)$ and $(\bar{\mathbf{t}}_1, \bar{\mathbf{t}}_1)$ are two linearly independent tangent directions to g and h respectively.

By eliminating the variables $u, v, w, \bar{u}, \bar{v}, \bar{w}$, and r, the closed-form of the Voronoi surface $\mathrm{Vor}(g, h)$ can be obtained in principle.

The closed-form of $\mathrm{Vor}(g, h)$ consists of two components, one is called the *even* Voronoi surface of g and h and the other is the *odd* Voronoi surface of g and h. The even Voronoi surface is the points that are either inside both g and h or outside both of them. The odd Voronoi surface consists of points that are inside one and outside the other basis surface. Both components arise when we cannot distinguish positive and negative offests. Notice that the elimination methods used to compute the closed-form of (1.5) cannot distinguish between even and odd Voronoi surfaces and hence the closed-form must be the union of two surfaces. Since $\mathrm{Vor}(g, h)$ is defined based on offset formulations, its closed-form may reveal extraneous factors which may be eliminated by adding additional equations: see [32]. To illustrate the above formulation, we give the following example.

Example 1.2 Let $g = x^2+(z-2)^2-1$ and $h = y^2+(z+2)^2-1$ be two cylinders of unit radius. Let r be the common radius of the spheres centering on g and h, and (x, y, z) be a point equidistant from g and h at (u, v, w) and $(\bar{u}, \bar{v}, \bar{w})$ respectively. At (u, v, w) on $g = 0$, $\mathbf{t}_1 = (0, 1, 0)$ and $\mathbf{t}_2 = (-(w-2), 0, u)$ are two linearly independent tangent vectors. At $(\bar{u}, \bar{v}, \bar{w})$ on $h = 0$, two linearly independent tangents are $\bar{\mathbf{t}}_1 = (1, 0, 0)$ and $\bar{\mathbf{t}}_2 = (0, -(\bar{w}+2), \bar{v})$. We then have the following system of eight equations representing $\mathrm{Vor}(g, h)$

$$\begin{aligned}
(x-u)^2 + (y-v)^2 + (z-w)^2 - r^2 &= 0 \\
u^2 + (w-2)^2 - 1 &= 0 \\
u(z-w) + (2-w)(x-u) &= 0 \\
y - v &= 0 \\
(x-\bar{u})^2 + (y-\bar{v})^2 + (z-\bar{w})^2 - r^2 &= 0 \\
\bar{u}^2 + (\bar{w}+2)^2 - 1 &= 0 \\
\bar{v}(z-\bar{w}) - (\bar{w}+2)(y-\bar{v}) &= 0 \\
x - \bar{u} &= 0
\end{aligned} \tag{1.6}$$

After eliminating $u, v, w, \bar{u}, \bar{v}, \bar{w}, r$ from (1.6), we obtain

$$(x^2 - y^2 - 8z)(48z^2 + 16y^2z - 16x^2z + y^4 - 2x^2y^2 - 8y^2 + x^4 - 8x^2 - 48)$$

where the first factor is the even Voronoi surface of g and h and the second one is the odd Voronoi surface. Note that the extraneous factors do not appear in the closed-form of $\mathrm{Vor}(g, h)$ due to the existence of two linearly independent tangents at every point of the cylinders. ■

Blending Surfaces Given two surfaces $g(x, y, z) = 0$ and $h(x, y, z) = 0$, a *blending surface* is a surface that intersects both surfaces tangentially along two curves. A *constant-radius* blend is a blending surface that has circular cross-sections of fixed radius. A *variable-radius* blend is a blend whose circular cross-sections are of variable radius. A constant-radius blend of g and h can be formulated as the envelope of the family of spheres of constant radius r whose centers are constrained to lie on $\mathrm{Offset}(g, r) \cap \mathrm{Offset}(h, r)$. Formalized in algebraic terms, it is the zero set of ten equations in twelve variables; see [30]. A variable-radius blend of g and h is the envelope of the family of spheres that have centers lying on $\mathrm{Vor}(g, h) \cap p$, where p is a reference surface, and have radii such that each sphere touches both g and h; see also [21, 30].

When the basis surface of the offset is in parametric form

$$x = x(u, v), \quad y = y(u, v), \quad z = z(u, v)$$

the algebraic formulation results in 3 equations in 5 variables as follows

$$\begin{aligned} (x - x(u,v))^2 + (y - y(u,v))^2 + (z - z(u,v))^2 - r^2 &= 0 \\ (x - x(u,v), y - y(u,v), z - z(u,v)) \cdot (x_u(u,v), y_u(u,v), z_u(u,v)) &= 0 \\ (x - x(u,v), y - y(u,v), z - z(u,v)) \cdot (x_v(u,v), y_v(u,v), z_v(u,v)) &= 0 \end{aligned} \tag{1.7}$$

where the subscripts denote partial differentiation. For Voronoi surfaces and blends involving parametric basis surfaces, the formulation is closely analogous to offsets.

This algebraic formulation of offsets, Voronoi surfaces and blends results in 2-surfaces in $\mathbf{R}^n$, $n > 3$, which are generally defined as the zero set of the following system of polynomial equations:

$$\begin{aligned} f_1(x_1, x_2, \ldots, x_n) &= 0 \\ f_2(x_1, x_2, \ldots, x_n) &= 0 \\ &\vdots \\ f_m(x_1, x_2, \ldots, x_n) &= 0 \end{aligned} \tag{1.8}$$

where $f_i \in K[x_1, x_2, \ldots, x_n]$ and m $(\geq n-2)$ is normally $n-2$. We denote system (1.8) in matrix form as $\mathbf{F}(\mathbf{x}) = \mathbf{0}$ and the zero set of $\mathbf{F}(\mathbf{x}) = \mathbf{0}$ as S_F. Recall that extra equations are sometimes required in order to eliminate extraneous factors introduced by linearly dependent tangent vectors in the offset formulation as shown, e.g., in (1.3). In the rest of the thesis, we assume $m = n-2$. For the cases of $m > n-2$ the modifications needed in the proposed computations are routine. It is worth remarking that the algebraic formulation of offsets, blends, and Voronoi surfaces given here all have the property that a 2-surface is defined in $\mathbf{R}^n$, where $n > 3$, but its projection into a certain subspace is wanted. In this thesis, we assume that the (x_1, x_2, x_3)-space is this subspace.

In principle, from system (1.8), the last $n-3$ variables of $\mathbf{x}$ can be eliminated. This computation reduces $\mathbf{F}(\mathbf{x}) = \mathbf{0}$ to a single equation

$$f(x_1, x_2, x_3) = 0 \tag{1.9}$$

in (x_1, x_2, x_3)-space with its zero set denoted as S_f. However, the elimination process is not practical. Hence the closed-form representation of $\mathbf{F}(\mathbf{x}) = \mathbf{0}$ in (x_1, x_2, x_3)-space, $f(x_1, x_2, x_3) = 0$, is often unobtainable in practice. In general S_f is the natural projection of S_F. Note that S_f might contain more points than the natural projection of S_F since the projection of S_F need not be an algebraic variety; see [69].

1.2 Implicit Approximations of Parametric Curves and Surfaces

A recurring operation in solid modeling is the evaluation of surface intersections [55]. If both surfaces are given parametrically, the two major approaches given the greatest prominence in the literature are subdivision and substitution methods.

In the subdivision method, e.g., [38, 40, 41, 42, 53], both surfaces are recursively subdivided in the vicinity of their intersection. The subdivision results in an adaptive piecewise linear approximation of both surfaces and their intersection. Among the advantages of the method we mention its robustness and its potential for locating all intersection branches. A major drawback of the subdivision method is the large volume of data it creates, which slows it down in areas of high surface curvature.

In the substitution method, e.g, [27, 44, 59, 67, 68], one of the surfaces, S_1, is converted to implicit form F, and the parametric form of S_2 is substituted into F resulting in an implicit algebraic curve f in the parameter space of S_2. This curve f is in birational correspondence with the intersection of S_1 and S_2 in xyz-space, and thus serves as an accurate representation of the intersection. Major difficulties of the substitution method limit its utility in practice. There are two general methods for implicitizing a parametric surface. The first method is based on elimination theory [61] and does resultant computations. It is expensive and generates extraneous factors whose detection is a delicate problem, see also Section 2.3. The second method for implicitization is based on Gröbner basis techniques [18]. It is also fairly expensive and requires, moreover, rational coefficients in the description of S_1. Another difficulty with the substitution method, less prominently pointed out but well-known [51], is that the substitution itself can be numerically unstable, and is a nontrivial algorithmic task when desiring efficiency and accuracy. Some authors have suggested the use of rational arithmetic for this reason [28], thus further adding to the computational load of the approach.

In Chapter 2, we provide a middle ground by deriving a local implicit approximation of rational or polynomial parametric curves and surfaces with low-degree implicit forms. In the context of subdivision techniques, such approximations have the potential of reducing the number of generated surface approximants, because we are not restricted to linear approximants only. In the context of substitution methods, the approximations avoid the high cost of implicitizing a parametric curve or surface, and provide, moreover, irreducible approximants.

Since the distribution of a preliminary version of [23], a number of related investigations have been developing and applying similar ideas. Bajaj and Ihm [15] apply a technique that is analogous to ours to the problem of designing blending surfaces and prove results on minimum degree blends satisfying certain constraints.

Previously, local *explicit* approximations to integral parametric curves and surfaces have been proposed in [48]. An approximant of the form

$$z = f(x,y) = \sum a_{ij}x^i y^j \quad \text{or} \quad y = f(x) = \sum a_i x^i$$

is constructed, for surfaces and curves. Recurrence formulas were also derived for the coefficients of f. Bajaj [11] extends this method using power series composition and inversion techniques together with rational Padé approximations. In our experience, a local explicit approximation is less favorable than a local implicit approximation. In fact, while a quadratic explicit approximation to a curve achieves second order contact at the point at which it is constructed, a quadratic implicit approximation achieves *fourth* order contact. For curves, the order of contact grows linearly with the degree of the explicit approximation, whereas the order of contact of the implicit approximation has a quadratic growth in the degree. Thus, much lower degree approximations suffice. Note, however, that for an implicit approximation of degree n, $O(n)$ coefficients can be chosen, whereas for a degree n explicit approximation only $O(n)$ coefficient are available.

In general, local explicit approximation can only approximate curves or surfaces *locally* no matter how high a degree of approximant is used. This is due

to the asymmetry introduced by making one variable an explicit function of the other(s). For instance, a circle cannot be completely approximated with a single explicit approximant. In contrast, our approximants are capable of approximating curves or surfaces not only locally but also globally in the sense that the radius of convergence increases when the degree of approximation increases, and the exact implicitization can be finally derived when the degree of approximation is equal to the degree of the given parametric curve or surface.

For a *properly parameterized* rational curve $\mathbf{r}(t)$, e.g., [60, 62], of degree m containing the origin, we seek an implicit curve $g(x,y) = 0$ of degree $n < m$ that approximates $\mathbf{r}(t)$ at the origin. The idea is to set up the polynomial $g(x,y)$ with symbolic coefficients e_{ij} and substitute $\mathbf{r}(t)$ into $g(x,y)$ with result

$$g(\mathbf{r}(t)) = \sum_{i=1}^{nm} \alpha_i t^i$$

where the α_i are linear combinations of the e_{ij}. We require that a certain number of the α_i vanish. With one of the coefficients being set to one and

$$\alpha_1 = 0, \;\; \alpha_2 = 0, \;\; \ldots, \;\; \alpha_s = 0$$

for some s the e_{ij} are determined and an implicit approximation is obtained that has contact of order s with $\mathbf{r}(t)$ at the origin.

We have derived a recurrence for computing the α_i directly from $\mathbf{r}(t)$ without explicit substitutions and shown that, when $n < m$,

- the coefficient matrix of $\alpha_1, \alpha_2, \ldots, \alpha_{nm}$ has rank that is equal to the number of unknown coefficients in $g(x,y)$.

- the degree n local implicit approximation $g(x,y)$ of $\mathbf{r}(t)$ at the origin is irreducible when the origin is a regular curve point.

The method derived here has the following merits compared to local explicit approximations proposed in [48, 11]:

- It works for polynomially and rationally parameterized curves and surfaces.
- It yields meaningful results for many types of singularity.
- The order of contact grows quadratically for curves and faster than linear for surfaces.
- The implicit approximants approximate curves and surfaces not only locally but also globally.

The algorithm for computing the implicit approximation of a properly parameterized rational surface is basically similar to the one for the curve case except that in the surface case additional safeguards are incorporated to ensure irreducibility.

1.3 Local Approximations of 2-D Surfaces

As described in Section 1.1.3, certain surfaces, including offset, blending and Voronoi surfaces, cannot be easily defined in the conventional 3-D space. In contrast, these surfaces can be formulated mathematically, with the theory of envelopes, in higher dimensional space in a straightforward manner. As the result, such surfaces are generally 2-dimensional surfaces in $\mathbf{R}^n$, $n > 3$, and are defined by (1.8) or in a matrix form $\mathbf{F}(\mathbf{x}) = \mathbf{0}$. Although the exact closed-form representation of such a surface $f(x_1, x_2, x_3) = 0$ could be derived in principle by elimination methods such as Gõbner basis [18, 19] or resultant techniques [46, 61, 13], it is often not feasible to do so due to the high complexity of these methods. Thus, as mentioned in [32], the viable approaches to interrogating such surfaces seem to be (a) approximate the surface locally and interrogate the approximant, or (b) interrogate the higher-dimensional representation. In Chapter 3, we have developed computational schemes for the local geometry of $f(x_1, x_2, x_3) = 0$ and have investigated techniques that construct implicit approximants, explicit approximants, and parametric approximants to the surface in (x_1, x_2, x_3)-space described by a system

of algebraic equations (1.8). These approximants have the forms $x_3 = \psi(x_1, x_2)$, and $(x_1 = \phi_1(s,t), x_2 = \phi_2(s,t), x_3 = \phi_3(s,t))$, respectively.

Since f is unknown and its derivation is often impractical, methods of computing the normal, tangent, and normal curvatures of $f = 0$ at a surface point in $\mathbf{R}^3$ from the information depending only on system (1.8) have been developed. Moreover, using a variant of the Three Tangent Theorem given by Pegna and Wolter [52], a method has been developed for computing a degree two local implicit approximant of $f = 0$.

The implicit function theorem ensures that system (1.8) determines $m = n - 2$ components of $\mathbf{x} = (x_1, \ldots, x_n)$ as functions of the remaining 2 components in a neighborhood of any surface point $\mathbf{x}^0$ on which the differential $D\mathbf{F}(\mathbf{x}^0)$ has rank $n - 2$. The unknown coefficients of explicit functions can be calculated by means of the chain rule and a linear system solver; however, it is algebraically tedious to do so. When we assume that $\mathbf{x}^0$ is the origin, a recursive formula has been derived which presents the computation in a more convenient manner. Thus, the recursive formula computes, without loss of generality, $x_i = \psi_i(x_1, x_2)$, $i = 3, \ldots, n$. It is clear that $x_3 = \psi_3(x_1, x_2)$ is a local explicit approximation of $f = 0$.

For a given system (1.8) and a regular point $\mathbf{x}^0$ on it, there exists a neighborhood of $\mathbf{x}^0$ in which the *parametrically defined solution* $x_i = \phi_i(u,v), i = 1, 2, \ldots, n$, can be found such that $\mathbf{x}^0 = (\phi_1(0,0), \ldots, \phi_n(0,0))$. The first three coordinate functions $(\phi_1(u,v), \phi_2(u,v), \phi_3(u,v))$ so computed constitute therefore a local parametrization of $f = 0$. To compute the parametric solution, we substitute symbolically x_i with $\phi_i(u,v)$, $i = 1, \ldots, n$, in all polynomials $f_i, i = 1, \ldots, n-2$. Then we compute the Taylor expansion of the resulting polynomials in u and v. By requiring that the coefficient of $u^j v^k$ is identically zero for $1 \leq j + k \leq l$, a series of linear systems are obtained. The solutions of the linear systems define a degree l approximation of the parametrically defined solution. The first three

component functions $x_1 = \varphi_1(u,v), x_2 = \varphi_2(u,v), x_3 = \varphi_3(u,v)$ so obtained comprise the degree l parametric approximant of $f = 0$. This derivation is analogous to the derivation of approximants of space curve proposed in [14].

1.4 Piecewise Linear Approximations of 2-D Surfaces

With the availability of piecewise linear approximations of surfaces, certain computations become practical; in particular, surface rendering can take advantage of hardware capabilities and we do not have to resort to expensive ray casting for visualization purposes. Based on subdivision, the PLA of major parametric surfaces in CAGD, including Bernstein-Bézier surface and B-spline surfaces, have been extensively studied and used with great success in surface interrogations [10, 16, 38, 41, 42]. However, it seems that much less research has been done in the literature to the PLA of implicitly defined surfaces [17, 9, 7, 8, 56, 57, 12].

Bloomenthal [17] has proposed an algorithm for computing PLA of an implicit surface $g(x,y,z) = 0$ based on space subdivision using octrees. The algorithm starts with an octree that bounds the surface portion of interest, and then decomposes recursively those octrees that intersects the surface until the surface portion inside the octree is sufficiently close to a plane. The surface function is evaluated at the corners of an octree cube, and from these values is determined the point at which the surface intersects an edge of the cube, by linear interpolation. The surface points on the edges of each cube are ordered to form a convex polygon each of whose edges lies in a cube face. One disadvantage of the vertex evaluation strategy is that the negatively signed corners of a cube cannot be separated from the positively signed corners by a single plane in all cases. Thus, vertex evaluation on a cube may result in ambiguities that could produce more than one polygonal approximation. In this case, further subdivision is usually required but does not necessarily resolve the ambiguity. Due to the nature of space decomposition,

this approach is capable of approximating all the surface components provided the initial cube has been properly situated.

In [9, 7, 8, 6], a simplicial continuation algorithm is presented for obtaining a PLA to a component of an implicitly defined 2-surface in $\mathbf{R}^n$. The algorithm starts at a regular surface point $\mathbf{x}^0$ and generates n-simplices in a sequential ordering which spiral outward from $\mathbf{x}^0$. The approximate zero points on the $(n-2)$-faces of each n-simplex are computed by an affine map and can be further refined using Newton iteration. Note that a n-simplex $\sigma = [\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_n]$ is the set of points in $\mathbf{R}^n$ that is the convex combination of $\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_n$. A n-simplex σ is said to be *transversal* if, roughly speaking, there exists an $(n-2)$-face with which the 2-surface intersects transversely. The standard *vector labeling* of $\mathbf{v} \in \mathbf{R}^n$ induced by $\mathbf{F}$

$$l_{\mathbf{F}}(\mathbf{v}) = \begin{bmatrix} 1 \\ \mathbf{F}(\mathbf{v}) \end{bmatrix}$$

is used to formally define transversality as follows. An $(n-2)$-face $[\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_{n-2}]$ is said to be *completely labeled* with respect to the vector labeling $l_{\mathbf{F}}$ if the *labeling matrix*

$$\Lambda = \begin{bmatrix} 1 & \cdots & 1 \\ \mathbf{F}(\mathbf{v}_0) & & \mathbf{F}(\mathbf{v}_{n-2}) \end{bmatrix}$$

has a lexicographically positive inverse, i.e., the first nonzero element in each row of Λ^{-1} is positive. An n-simplex σ is transversal if there exists an $(n-2)$-face $\tau \subset \sigma$ which is completely labeled with respect to the vector labeling $l_{\mathbf{F}}$.

For a n-simplex, the affine map $\mathbf{H}_\sigma : \mathbf{R}^n \to \mathbf{R}^{n-2}$ is a linear map which interpolates $\mathbf{F}(\mathbf{x})$ on the vertices of σ, i.e., $\mathbf{H}_\sigma(\mathbf{v}_i) = \mathbf{F}(\mathbf{v}_i)$ for $i = 0, 1, \ldots, n$. When σ is transversal $\sigma \cap \mathbf{H}_\sigma^{-1}(0)$ serves as an approximation to the 2-surface inside σ.

Let $\mathbf{T}$ represent the collection of all transversal simplices in a compact domain and the piecewise linear approximation $\mathbf{H}_\mathbf{T}$ of $\mathbf{F}(\mathbf{x})$ relative to $\mathbf{T}$ be defined as $\mathbf{H}_\sigma$

when restricted to σ for every $\sigma \in \mathbf{T}$. Then,

$$\mathbf{H}_{\mathbf{T}}^{-1}(0) = \{\, \mathbf{H}_{\sigma}^{-1}(0) \mid \sigma \in \mathbf{T} \}$$

is the PLA of the surface component that contains $\mathbf{x}^0$. Newton iteration can be used to refine points on $\mathbf{H}_{\mathbf{T}}^{-1}(0)$ to surface points. An error bound for $\|\mathbf{F}(\mathbf{x})\|$, where $\mathbf{x} \in \mathbf{H}_{\mathbf{T}}^{-1}(0)$, in terms of the mesh size is provided and it is shown that $\mathbf{H}_{\mathbf{T}}^{-1}(0)$ approximates $\mathbf{F}^{-1}(0)$ quadratically in the mesh size [5].

While Bloomenthal's method is more suited to implicit surfaces in $\mathbf{R}^3$, the simplicial continuation method is designed for general 2-surfaces in $\mathbf{R}^n$. However, the direct application of these methods to computing PLA of offsets, Voronoi surfaces, and blends will have a space and time complexity that is exponential in the dimensionality of the space.

Rheinboldt [56, 57] has presented a continuation method that maps a reference triangulation Σ on $\mathbf{R}^p$ to a p-manifold M in $\mathbf{R}^n$, $n > p \geq 1$, and hence produces a triangulation on M. The method consists of two major computation schemes. A moving frame algorithm is given to derive orthonormal bases, i.e. local coordinate systems of the tangent space $T_x(M)$, that vary continuously with their point of contact x on M. A triangulation algorithm uses the orthonormal bases produced by the moving frame algorithm to map the vertices of the reference triangulation onto the tangent space $x + T_x(M)$ corresponding to appropriate points x on M. Thereafter, Gauss-Newton iteration is applied to *project* these triangulations from $x + T_x(M)$ onto M. At each step of the triangulation algorithm, a reference vertex ξ of the reference triangulation Σ and the corresponding point $x \in M$ are selected. The *center* ξ is associated with a set $\Gamma(\xi)$ of vertices of Σ that can be mapped onto M. Then the reference vertices in $\Gamma(\xi)$ that have not been processed are mapped onto $x + T_x(M)$ and projected onto M by the Gauss-Newton iteration. To proceed, a reference vertex in $\Gamma(\xi)$ that has been processed is selected as the next ξ and the same computation is applied. The points computed on M inherit the connectivity structure of Σ which in turn induces a triangulation on M. Note that

the triangulations induced by two overlapped $\Gamma(\xi_1)$ and $\Gamma(\xi_2)$ are made compatible by demanding that the reference vertices in $\Gamma(\xi_1) \cap \Gamma(\xi_2)$ are processed only once.

In Chapter 4, we propose an algorithm that is capable of computing PLA for 2-surfaces projected into $\mathbf{R}^3$ but defined algebraically in $\mathbf{R}^n$, $n > 3$.

1.5 Some Preliminaries

A rational plane curve $\mathbf{r}(t)$ can be given as the pair $(x(t), y(t))$, where $x(t)$ and $y(t)$ are rational functions of t. The curve points are all points $(x(t), y(t))$ on the plane. The curve is *properly parameterized* if for all but finitely many curve points p we have $p = (x(t), y(t))$ with a *unique* value of t. When a parametric curve is not properly parameterized, there exists a rational nonlinear function $s(t)$ such that $x(t) = x^*(s(t))$ and $y(t) = y^*(s(t))$ for some rational functions x^* and y^*. We assume in this thesis that all parametric curves are properly parameterized and note that a parametric curve is always irreducible. For methods to detect improper parameterization, see Sederberg [62].

The degree of a rational parametric curve is the highest degree of the numerator or the denominator polynomial, assuming both $x(t)$ and $y(t)$ have been written with a common denominator. The implicit equation $f(x, y)$ of the rational curve $\mathbf{r}(t)$ is a lowest degree polynomial in x and y satisfying $f(x(t), y(t)) \equiv 0$. It is unique up to a multiplicative constant. If $\mathbf{r}(t)$ has degree m, then so does $f(x, y)$; see, e.g., [46, 49].

As with parametric curves, a parametric surface

$$\mathbf{P}(s, t) = (x(s, t), y(s, t), z(s, t))$$

can be *improperly parameterized* if there are nonlinear rational functions $u(s, t)$ and $v(s, t)$ such that

$$\mathbf{P}(s, t) = (x^*(u(s, t), v(s, t)), y^*(u(s, t), v(s, t)), z^*(u(s, t), v(s, t)))$$

for some x^*, y^* and z^*. In that case, there is a many-to-one correspondence between the parameter values and the surface points. $\mathbf{P}(s,t)$ is *properly parameterized* if this correspondence is one-to-one except, possibly, on a one dimensional set of points. We also assume that all parametric surfaces are properly parameterized. For a parametric surface described by rational functions of total degrees m, there always exists an irreducible implicit equation $f(x,y,z) = 0$ satisfying $f(x(s,t), y(s,t), z(s,t)) \equiv 0$, and f is unique within a constant factor, see, e.g., [22]. Moreover, f has degree at most m^2.

A polynomial of degree k in the variables $x_1, x_2, ..., x_n$ is denoted $f^k(x_1, ..., x_n)$ whenever we wish to stress the degree. The *gradient* of $f = 0$ at the point $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ is the vector $\nabla f = (f_{x_1}, f_{x_2}, \ldots, f_{x_n})$, where the partials are evaluated at $\mathbf{x}$. The *Hessian* of f is the symmetric matrix

$$\begin{bmatrix} f_{x_1x_1} & f_{x_1x_2} & \cdots & f_{x_1x_n} \\ f_{x_2x_1} & f_{x_2x_2} & \cdots & f_{x_2x_n} \\ \vdots & \vdots & \vdots & \vdots \\ f_{x_nx_1} & f_{x_nx_2} & \cdots & f_{x_nx_n} \end{bmatrix}$$

where subscripts denote partial differential.

A point $\mathbf{x}^0 = (x_1^0, \cdots, x_n^0)$ is said to be *non-singular* or *regular* on f if the gradient of f at $\mathbf{x}^0$ is not null; otherwise the point is *singular*.

Given a system of algebraic equations (1.8), the $m \times n$ matrix

$$D\mathbf{F}(\mathbf{x}^0) \equiv \left(\frac{\partial f_i}{\partial x_j}\right)_{\mathbf{x}=\mathbf{x}^0}$$

is called the *differential* of $\mathbf{F}$ at $\mathbf{x}^0 = (x_1^0, \cdots, x_n^0)$. A point $\mathbf{x}^0$ is said to be *non-singular* or *regular* on the 2-surface $\mathbf{F}$ if $D\mathbf{F}(\mathbf{x}^0)$ has rank $n-2$. That is, the gradients $\nabla f_1, \ldots, \nabla f_m$ form a normal space of dimension $n-2$. When $m = n-2$, this is equivalent to saying that $\mathbf{x}^0$ is non-singular on all f_i, $i = 1, 2, \ldots, n-2$, and the gradients $\nabla f_1, \ldots, \nabla f_m$ are linearly independent. $\mathbf{x}^0$ is *singular* if it is not a non-singular point.

1.6 Thesis Organization

In this first chapter, various types of surface encountered in CAGD and solid modeling, including parametric surfaces, implicit surfaces, and 2-surfaces in high-dimensional space, have been presented, and the problems of approximating curves and surfaces and their applications to surface interrogations have been discussed. Moreover, several local and global approximation techniques that will be proposed in this thesis have been sketched.

In Chapter 2, a method for computing implicit approximations to parametric curves or surfaces is proposed that might circumvent the difficulty of globally implicitizing parametric curves or surfaces. Chapter 3 deals with the local geometry and local approximations of 2-surface in high-dimensional space. In Chapter 4, an algorithm is developed for computing the piecewise linear approximation of offsets, Voronoi surfaces, and blends. Chapter 5 provides a summary of this research and comments on some future research directions.

2. IMPLICIT APPROXIMATION OF CURVES AND SURFACES

A method is described for finding an implicit approximant to a parametric curve or surface in a neighborhood of a point. The method works for both polynomially and rationally parameterized curves and surfaces, and achieves an order of contact that can be prescribed. In the case of nonsingular curve points, the approximant must be irreducible, but in the surface case additional safeguards have been incorporated into the algorithm to ensure irreducibility. The method also yields meaningful results for many types of singularity. The chapter is organized as follows. In Section 2.1, we describe the method for polynomially and rationally parameterized curves. Section 2.2 presents the surface case. In Section 2.3, we comment briefly on some theoretical connections between the method we propose here and several resultant formulations found in the literature.

2.1 Local Implicit Approximation of Parametric Plane Curves

We seek an implicit curve $g(x, y) = 0$ that approximates the parametric curve $\mathbf{r}(t) = (x(t), y(t))$ at the origin subject to a prescribed order of contact. The idea is to set up a polynomial $g(x, y)$ of sufficiently high degree with symbolic coefficients e_{ij}. Then, a system of linear equations with unknowns e_{ij} is formulated and solved.

The linear system is obtained by substituting $\mathbf{r}(t)$ into $g(x, y)$. The result is

$$g(x(t), y(t)) = \sum \alpha_k t^k$$

where the α_k are linear combinations of the e_{ij}. We require that a certain number of the α_k vanish. With

$$\alpha_1 = 0, \ \alpha_2 = 0, \ \ldots, \ \alpha_s = 0$$

for some s, an implicit approximation is obtained that has contact of order s with $\mathbf{r}(t)$ at the origin. The approach depends on the following details:

1. There is a recurrence for deriving the linear system directly from $\mathbf{r}(t)$ without explicit substitutions. This recurrence is derived in Section 2.1.1.

2. Assume that the degree n of the approximation is smaller than m, the degree of $\mathbf{r}(t)$. There is a function $\varphi(n)$ that determines the order of contact that $g(x, y)$ can achieve. This function is obtained by analyzing the rank of the linear system in Section 2.1.2.

In Section 2.1.3 we discuss the error behavior of the implicit approximation, and, in Section 2.1.4, we present several experiments.

Let

$$\mathbf{r}(t) = (x(t), y(t)) = (\frac{p(t)}{w(t)}, \frac{q(t)}{w(t)})$$

be a properly parameterized rational curve of degree m containing the origin, where

$$p(t) = \sum_{i=1}^{m} a_i t^i, \qquad q(t) = \sum_{i=1}^{m} b_i t^i, \qquad w(t) = \sum_{i=0}^{m} c_i t^i$$

We assume that a_m and b_m are not both zero, and that $c_0 \neq 0$. From [61, 63], we know that there exists an irreducible polynomial $f^m(x, y) = 0$ of degree m such that

$$f^m(x(t), y(t)) \equiv 0$$

Let $g^n(x, y) = \sum_{i+j=1}^{n} e_{ij} x^i y^j = 0$ be a degree n implicit curve containing the origin. Since $g^n(x, y) = 0$ and $\gamma g^n(x, y) = 0$, where $\gamma \neq 0$, are the same curve, $g^n(x, y) = 0$ has $\varphi(n) = (n^2 + 3n - 2)/2$ coefficients on which the curve depends. Let $G^n(x, y, z)$ be the homogeneous form of $g^n(x, y)$. Substitution yields

$$g^n(\frac{p(t)}{w(t)}, \frac{q(t)}{w(t)}) = \frac{G^n(p(t), q(t), w(t))}{(w(t))^n} = \frac{\sum_{i=1}^{nm} \alpha_i t^i}{(w(t))^n}$$

where the α_i are linear combinations of the e_{ij}.

We look for an implicit form $g^n(x,y) = 0$ of degree $n < m$ approximating $\mathbf{r}(t)$ at the origin, and the method of deriving should work whether $w(t) \equiv 1$ or not. i.e., irrespective whether the curve is parameterized polynomially or rationally. The following simple example demonstrates the approach.

Example 2.1 Consider $\mathbf{c}_0(t) = (x(t), y(t)) = (p(t)/w(t), q(t)/w(t))$, where $p(t) = 2t^3 + t^2 - 3t$, $q(t) = t^3 - t^2 - 2t$, and $w(t) = t^2 + 4t + 5$. $\mathbf{c}_0(t)$ is a properly parameterized plane curve containing the origin. Let $g^2(x,y) = e_{10}x + e_{01}y + e_{20}x^2 + e_{11}xy + e_{02}y^2$ be a degree 2 curve containing the origin with symbolic coefficients. Substituting $x(t)$ and $y(t)$ into g^2 yields $g^2(x(t), y(t)) = (\sum_{i=1}^{6} \alpha_i t^i)/(w(t))^2$ where

$$\begin{aligned}
\alpha_1 &= -15e_{10} - 10e_{01} \\
\alpha_2 &= -7e_{10} - 13e_{01} + 9e_{20} + 6e_{11} + 4e_{02} \\
\alpha_3 &= 11e_{10} - e_{01} - 6e_{20} + e_{11} + 4e_{02} \\
\alpha_4 &= 9e_{10} + 3e_{01} - 11e_{20} - 8e_{11} - 3e_{02} \\
\alpha_5 &= 2e_{10} + e_{01} + 4e_{20} - e_{11} - 2e_{02} \\
\alpha_6 &= 4e_{20} + 2e_{11} + e_{02}
\end{aligned}$$

By requiring $e_{10} - 1 = 0, \alpha_1 = 0, \alpha_2 = 0, \alpha_3 = 0$, and $\alpha_4 = 0$, we can solve for the unknown coefficients e_{ij}. The resulting g^2 approximates $\mathbf{c}_0$ at the origin with contact of order four. ■

2.1.1 A Recurrence for α_k

Since $g^n(x,y) = g^{n-1}(x,y) + \sum_{i+j=n} e_{ij}x^i y^j$, the homogeneous form of $g^n(x,y)$ can be written as

$$G^n(x,y,z) = zG^{n-1}(x,y,z) + \sum_{i+j=n} e_{ij}x^i y^j \tag{2.1}$$

In the following, let α_k^{n-1} and α_k^n denote the coefficient of t^k in $G^{n-1}(p(t), q(t), w(t))$ and $G^n(p(t), q(t), w(t))$, respectively. It is clear that α_k^n can be derived from the $\alpha_i^{n-1}, i = 1, 2, \ldots, k$, because of (2.1).

We define $(a(i))_l$ and $(b(j))_l$ as in [48], setting

$$(p(t))^i = (\sum_{l=1}^{m} a_l t^l)^i = \sum_{l=i}^{im} (a(i))_l t^l$$

and similarly,

$$(q(t))^j = (\sum_{l=1}^{m} b_l t^l)^j = \sum_{l=j}^{jm} (b(j))_l t^l$$

A recurrence to compute the $(a(i))_l$ and $(b(j))_l$ is derived as follows, see also [48]. By definition of $(a(i+1))_l$, it follows that

$$\begin{aligned} \sum_{l=i+1}^{(i+1)m} (a(i+1))_l x^l &= \left(\sum_{j=i}^{im} (a(i))_j t^j\right)\left(\sum_{k=1}^{m} a_k t^k\right) \\ &= \sum_{j=i}^{im}\sum_{k=1}^{m} (a(i))_j a_k t^{j+k} \end{aligned}$$

Setting $l = j + k$ and $p = j$ yields $i \le p \le im$ and $1 \le l - p = k \le m$, and

$$\sum_{l=i+1}^{(i+1)m} (a(i+1))_l x^l = \sum_{l=i+1}^{(i+1)m} \left(\sum_{p=i}^{l-1} (a(i))_p a_{l-p}\right) t^l$$

Thus, equating coefficients yields the recurrence

$$(a(i+1))_l = \sum_{p=i}^{l-1} (a(i))_p a_{l-p}$$

Recurrence for $(b(j)_l$ is analogous.

From (2.1), we therefore obtain

$$\begin{aligned} \alpha_k^n &= \textit{coefficient of } t^k \textit{ in } (w(t) \sum_{j=1}^{m(n-1)} \alpha_j^{n-1} t^j + \sum_{i+j=n} e_{ij} (p(t))^i (q(t))^j) \\ &= \sum_{j=1}^{k} \alpha_j^{n-1} c_{k-j} + \sum_{i+j=n} \sum_{p+q=k} e_{ij} (a(i))_p (b(j))_q \end{aligned}$$

In particular, $\alpha_k^1 = e_{10} a_k + e_{01} b_k$.

For an integral parametric curve $\mathbf{r}(t)$, a straightforward computation shows that the α_k^n specialize to

$$\alpha_k^n = \begin{cases} \alpha_k^{n-1} & 1 \le k \le n-1 \\ \alpha_k^{n-1} + \sum_{i+j=n} \sum_{p+q=k} e_{ij} (a(i))_p (b(j))_q & n \le k \le (n-1)m \\ \sum_{i+j=n} \sum_{p+q=k} e_{ij} (a(i))_p (b(j))_q & (n-1)m < i \le nm \end{cases}$$

2.1.2 Derivation of the Method

2.1.2.1 Rank of the Linear System

Having explained how to obtain the α_k^n, we now show that the coefficient matrix of the linear system defined by setting $\alpha_k^n = 0, k = 1, 2, \ldots, nm$, has rank at least $\varphi(n)$. We are able to determine a nontrivial solution to unknown coefficients by setting one of the coefficient to 1 and solving the system $\alpha_1^n = 0, \alpha_2^n = 0, \ldots, \alpha_s^n = 0$ for $s \geq \varphi(n)$ chosen such that the rank is $\varphi(n)$.

Let $\mathbf{e}_n = (e_{10}, e_{01}, e_{20}, e_{11}, e_{02}, \ldots, e_{n0}, e_{(n-1)\,1}, \ldots, e_{1\,(n-1)}, e_{0n})^T$ be the vector of unknowns, and write the system of equations $\alpha_1^n = 0, \alpha_2^n = 0, \ldots, \alpha_{nm}^n = 0$ in matrix form:

$$\mathbf{A}_{mn}\mathbf{e}_n = 0 \tag{2.2}$$

Note that $\mathbf{A}_{mn}$ is a nm by $\varphi(n) + 1$ matrix. Furthermore, the maximum rank of $\mathbf{A}_{mn}$ is $\varphi(n) + 1$ since $m \geq n$ and $nm \geq \varphi(n) + 1$. Example 3.2 shows matrix $\mathbf{A}_{32}$ symbolically.

Example 2.2 For $m = 3$ and $n = 2$, $\mathbf{A}_{32}$ is

$$\begin{bmatrix} a_1c_0 & b_1c_0 & 0 & 0 & 0 \\ a_1c_1 + a_2c_0 & b_1c_1 + b_2c_0 & a_1^2 & a_1b_1 & b_1^2 \\ a_1c_2 + a_2c_1 + a_3c_0 & b_1c_2 + b_2c_1 + b_3c_0 & 2a_1a_2 & a_1b_2 + a_2b_1 & 2b_1b_2 \\ a_1c_3 + a_2c_2 + a_3c_1 & b_1c_3 + b_2c_2 + b_3c_1 & 2a_1a_3 + a_2^2 & a_1b_3 + a_2b_2 + a_3b_1 & 2b_1b_3 + b_2^2 \\ a_2c_3 + a_3c_2 & b_2c_3 + b_3c_2 & 2a_2a_3 & a_2b_3 + a_3b_2 & 2b_2b_3 \\ a_3c_3 & b_3c_3 & a_3^2 & a_3b_3 & b_3^2 \end{bmatrix}$$

■

When computing the local implicit approximation $g^n(x, y)$ of $\mathbf{r}(t)$, if the rank of $\mathbf{A}_{mn}$ is at least $\varphi(n)$ then we can select one coefficient of $g^n(x, y)$ to be 1 and determine the others by selecting the first s rows of (2.2) and choosing s such that the system has rank $\varphi(n)$

Let $f^m(x,y)=0$ be the exact irreducible implicit form of $\mathbf{r}(t)$ with $F^m(x,y,z)$ as its corresponding homogeneous form, and let f^n, $n<m$, be the degree n *initial segment* of f^m with its corresponding homogeneous form $F^n(x,y,z)$, that is, $f^m(x,y)=f^n(x,y)+$*terms with degree* $>n$. Also, let $\sum_{i=1}^{nm}\bar{b}_i t^i \equiv F^n(p(t),q(t),w(t))$ and $\bar{\mathbf{b}}_{mn}\equiv(\bar{b}_1,\bar{b}_2,\ldots,\bar{b}_{mn})^T$.

Lemma 2.1

1. *$f^n(x,y)$ is the zero polynomial if and only if $\bar{\mathbf{b}}_{mn}=\mathbf{0}$.*
2. *If $f^n(x,y)$ is a nonzero polynomial then $\bar{b}_1=\bar{b}_2=\cdots=\bar{b}_n=0$ and $\bar{\mathbf{b}}_{mn}\neq\mathbf{0}$.*

Proof: part 1: "$\Rightarrow$" trivial. "$\Leftarrow$" Suppose $f^n(x,y)$ is a nonzero polynomial, since $\bar{\mathbf{b}}_{mn}=\mathbf{0}$.

$$F^n(p(t),q(t),w(t))=0=(w(t))^n f^n(\frac{p(t)}{w(t)},\frac{q(t)}{w(t)})$$

for all t and then $f^n(p(t)/w(t),q(t)/w(t))=0$, for all t with possibly finitely many exceptions, where $w(t)=0$. Thus $f^n(x,y)$ with $n<m$ also represents $\mathbf{r}(t)$ which contradicts the irreducibility of $f^m(x,y)$.

part 2: Since $f^m(x,y)=0$ is the implicit form of $\mathbf{r}(t)$,

$$f^m(\frac{p(t)}{w(t)},\frac{q(t)}{w(t)})=0$$

for all t except finitely many t where $w(t)=0$. Thus

$$F^m(p(t),q(t),w(t))=(w(t))^m f^m(\frac{p(t)}{w(t)},\frac{q(t)}{w(t)})=0$$

for all t. From

$$F^n(p(t),q(t),w(t))+\sum_{i+j=n+1}^{m} e_{ij}(p(t))^i(q(t))^j(w(t))^{m-i-j}=0$$

for every t, we have

$$\sum_{i=1}^{nm}\bar{b}_i t^i=-\sum_{i+j=n+1}^{m} e_{ij}(p(t))^i(q(t))^j(w(t))^{m-i-j}=\sum_{i=n+1}^{nm}\bar{b}'_i t^i$$

for every t. By comparison, we have $\bar{b}_1 = \bar{b}_2 = \cdots = \bar{b}_n = 0$. The rest of part 2 follows from part 1. ■

Since $f^n(x,y)$ is an initial segment of $f^m(x,y)$, it could be either a zero polynomial or a nonzero polynomial with zero or nonzero $\bar{\mathbf{b}}_{mn}$ vectors respectively. If $\bar{\mathbf{b}}_{mn}$ is known beforehand, the coefficient vector $\mathbf{e}_n$ of $f^n(x,y)$ is uniquely determined by $\mathbf{A}_{mn}\mathbf{e}_n = \bar{\mathbf{b}}_{mn}$ which is an overdetermined linear system. Note that, for a fixed n, the elements of the matrix $\mathbf{A}_{mn}$ depend only on the coefficients of $p(t), q(t)$, and $w(t)$. The following results characterize the rank of $\mathbf{A}_{mn}$.

Lemma 2.2 *If* $\mathbf{r}(t)$ *is a properly parameterized rational plane curve of degree* m *then for* $n < m$, *we have*

$$rank(\mathbf{A}_{mn}) = \varphi(n) + 1$$

Proof: Suppose, knowing $\bar{\mathbf{b}}_{mn}$, we want to determine the coefficients of $f^n(x,y)$ by solving the overdetermined linear system $\mathbf{A}_{mn}\mathbf{e}_n = \bar{\mathbf{b}}_{mn}$, where $\mathbf{e}_n$ is the coefficient vector of the general degree n polynomial. Since $\mathbf{A}_{mn}$ depends on the coefficients of $\mathbf{r}(t)$, we consider the following two cases.

First, if $\mathbf{r}(t)$ is a curve such that the corresponding $f^n(x,y)$ is the zero polynomial, then $\bar{\mathbf{b}}_{mn} = \mathbf{0}$ and $\mathbf{A}_{mn}\mathbf{e}_n = \mathbf{0}$ is a homogeneous system. If $rank(\mathbf{A}_{mn}) < \varphi(n) + 1$, there will be infinitely many nontrivial solutions as well as the trivial solution for this linear system. This cannot be true by Lemma 2.1. Thus $rank(\mathbf{A}_{mn}) = \varphi(n) + 1$.

Second, if $\mathbf{r}(t)$ is a curve such that the corresponding $f^n(x,y)$ is a nonzero polynomial, then $\bar{\mathbf{b}}_{mn} \neq \mathbf{0}$ and $\mathbf{A}_{mn}\mathbf{e}_n = \bar{\mathbf{b}}_{mn}$ is a *consistent* non-homogeneous system since there is always a solution, that is with e_{ij} the coefficients of $f^n(x,y)$. Suppose $rank(\mathbf{A}_{mn}) < \varphi(n)+1$, this system will have infinitely many solutions. Let $\mathbf{e}_n^*$ be one of the infinitely many solutions and $\mathbf{e}_n^* \neq \mathbf{e}_n$, where $\mathbf{e}_n$ is the coefficient vector of $f^n(x,y)$. Let also $h^n(x,y)$ be the corresponding polynomial of $\mathbf{e}_n^*$ and

$$h^m(x,y) \equiv h^n(x,y) - terms\ of\ f^m(x,y)\ with\ degree > n$$

Let $H^m(x,y,z)$ and $H^n(x,y,z)$ be the homogeneous polynomials of $h^m(x,y)$ and $h^n(x,y)$ respectively. Since $\mathbf{A}_{mn}\mathbf{e}_n = \mathbf{A}_{mn}\mathbf{e}_n^* = \bar{\mathbf{b}}_{mn}$,

$$H^n(p(t),q(t),w(t)) = F^n(p(t),q(t),w(t))$$

for every t, and thus

$$H^m(p(t),q(t),w(t)) = F^m(p(t),q(t),w(t)) = 0$$

for all t. We then have

$$f^m(\frac{p(t)}{w(t)},\frac{q(t)}{w(t)}) = h^m(\frac{p(t)}{w(t)},\frac{q(t)}{w(t)}) = 0$$

for all but finitely many t. Hence $f^m(x,y)$ and $h^m(x,y)$ represent the same algebraic curve. Since $f^n(x,y) \neq h^n(x,y)$, $f^m(x,y)$ and $h^m(x,y)$ differ by more than a constant factor which contradicts that the equation of an irreducible curve is unique to within a constant factor. Therefore $rank(\mathbf{A}_{mn})$ must be $\varphi(n)+1$. ■

Lemma 2.3 *If* $\mathbf{r}(t)$ *is a properly parameterized rational plane curve of degree* m *then for* $n = m$, *we have*

$$rank(\mathbf{A}_{mm}) = \varphi(m)$$

Proof: If $n = m$, $f^m(p(t)/w(t), q(t)/w(t)) = 0$ for all t except finitely many t where $w(t) = 0$, and then $F^m(p(t), q(t), w(t)) = 0$ for all t, we thus have $\mathbf{A}_{mm}\mathbf{e}_m = \mathbf{0}$ which is an overdetermined linear homogeneous system. Since we will have only trivial solution if $rank(\mathbf{A}_{mm}) = \varphi(m)+1$, $rank(\mathbf{A}_{mm})$ must be less than or equal to $\varphi(m)$.

Suppose $r \equiv rank(\mathbf{A}_{mm}) < \varphi(m)$, then the solution space of the overdetermined homogeneous system has as basis $p \equiv \varphi(m)+1-r$ linearly independent vectors and every solution of this system is the linear combination of these p solutions. Now suppose that $r < \varphi(m)$, then the system has a solution space spanned by $p > 2$ linearly independent vectors, say $\mathbf{e}_m^1, \mathbf{e}_m^2, \cdots, \mathbf{e}_m^p$. Let $f_i^m(x,y)$ be the

corresponding polynomial with coefficient vector $\mathbf{e}_m^i$ and $F_i^m(x,y,z)$ be the homogeneous form of $f_i^m(x,y)$, $i = 1,2,\ldots,p$. Since

$$F_i^m(p(t),q(t),w(t)) = 0 = (w(t))^m f_i^m(\frac{p(t)}{w(t)},\frac{q(t)}{w(t)})$$

for all t, $1 \leq i \leq p$, we have $f_i^m(p(t)/w(t),q(t)/w(t)) = 0$ for all t with finitely many exceptions where $w(t) = 0$, $1 \leq i \leq p$. Thus the irreducible curve $f^m(x,y) = 0$ can be represented by $f_i^m(x,y), i = 1,2,\ldots,p$, which are not different within only a constant factor because $\mathbf{e}_m^1, \mathbf{e}_m^2, \cdots, \mathbf{e}_m^p$ are linearly independent. By the above arguments, we can conclude that $rank(\mathbf{A}_{mm}) = \varphi(m)$. ■

By assigning one variable to be 1, the existence of a nontrivial solution of $\mathbf{A}_{mm}\mathbf{e}_m = \mathbf{0}$ is guaranteed by Lemma 2.3, and it is the coefficient vector of the exact implicitization of $\mathbf{r}(t)$.

Observe that Lemmas 2.2 and 2.3 are not valid for *improperly parameterized* rational plane curves, as shown in the following example:

Example 2.3 Let $\mathbf{c}_1(t) = (x(t),y(t)) = (t^2+2t, t^4+4t^3+6t^2+4t)$. Since $x(t) = s(t)$ and $y(t) = (s(t))^2$, where $s(t) = t^2+2$, $\mathbf{c}_1(t)$ is improperly parameterized. For $n = 2$ the rank of $\mathbf{A}_{42}$ is 4 whereas $\varphi(2)+1$ is 5. ■

We summarize the lemmas above in the following

Theorem 2.1 *If $\mathbf{r}(t)$ is a properly parameterized rational plane curve of degree m then*

$$rank(\mathbf{A}_{mn}) = \begin{cases} \varphi(n)+1 & \text{if } n < m \\ \varphi(n) & \text{if } n = m \end{cases}$$

where n is the degree of the approximant, $\varphi(n) = (n^2+3n-2)/2$ is the number of coefficients on which the approximant depends.

2.1.2.2 The Algorithm

Because of Theorem 2.1, we may compute the degree n local implicit approximation as follows:

Let

$$\mathbf{B}\mathbf{e}_n = \mathbf{0} \tag{2.3}$$

be the subsystem of (2.2) consisting of the first s equations of $\mathbf{A}_{mn}\mathbf{e}_n = \mathbf{0}$ such that $\mathbf{B}$ has rank $\varphi(n)$. If the origin is not a singular curve point, then augment the system (2.3) with an equation $e_{01} = 1$ or $e_{10} = 1$ according to whether $x'(0) \neq 0$ or $y'(0) \neq 0$. If the origin is a singular curve point, on the other hand, then (2.3) is augmented by the equation $e_{ij} = 1$ where the indices i and j are selected by inspecting the system. In this way, a linear system

$$\mathbf{C}\mathbf{e}_n = \mathbf{b} \tag{2.4}$$

is obtained that has a nontrivial solution for $\mathbf{e}_n$. Since $\alpha_1^n = 0, \alpha_2^n = 0, \ldots, \alpha_s^n = 0$, the curve g^n so defined must have contact of degree at least s to $\mathbf{r}(t)$ at the origin.

There may be cases in which the system (2.4) is inconsistent, i.e., the augmented matrix $[\mathbf{C}, \mathbf{b}]$ is of rank $\varphi(n)+2$ while $\mathbf{C}$ has rank $\varphi(n)+1$. In this case, the linear system can be modified to ensure consistency. For instance, when computing g^2 of $\mathbf{c}_2(t) = (t, t^9)$, α_9^2 should be removed from, and $\alpha_{18}^2 = 0$ should be added to the system (2.4), resulting in $e_{01} - 1 = 0, \alpha_1^2 = 0, \alpha_2^2 = 0, \ldots, \alpha_8^2 = 0, \alpha_{10}^2 = 0, \alpha_{18}^2 = 0$. In this way a $g^2(x, y) = y$ is obtained, an approximation that has eighth order of contact and is irreducible.

2.1.2.3 Irreducibility of Implicit Approximations

When the origin is a regular curve point we show that the implicit approximation $g^n(x, y)$ of $\mathbf{r}(t)$ at the origin is irreducible whenever the linear system (2.4) is consistent. Note that the local implicit approximations of different degrees derived have the same linear terms if the equations augmented to (2.3) are the same. In the following lemmas, we assume that (2.4) is a consistent system. Also, let $s(n)$ be the order of contact made by the degree n implicit approximation $g^n(x, y)$.

Lemma 2.4 *If $g^n(x,y)$ and $g^{n-1}(x,y)$ of $\mathbf{r}(t)$ at the nonsingular point $(0,0)$ are computed by augmenting the system (2.3) the same $\alpha = 0$, where $\alpha = 0$ is either $e_{10} - 1 = 0$ or $e_{01} - 1 = 0$, then we have $s(n-1) < s(n)$.*

Proof: Let $\bar{g}^n(x,y) = \frac{1}{2}(g^n(x,y)+g^{n-1}(x,y))$. Suppose $s(n-1) \geq s(n)$, the $\bar{g}^n(x,y)$ so defined has the following four properties: (1) $\bar{g}^n$ is of degree n. (2) $\bar{g}^n \neq g^n$. (3) $\bar{g}^n$ has the same linear terms as g^n since g^n and g^{n-1} have the identical linear terms. (4) $\bar{g}^n$ has the order of contact larger than or equal to $s(n)$ since $s(n-1) \geq s(n)$ is assumed. From properties 1, 3, and 4, the coefficients of $\bar{g}^n$ satisfy the linear system that is used to compute the coefficients of g^n; but property 2 contradicts the uniqueness of solution of a nonsingular linear system. Thus $s(n-1) < s(n)$. ■

By induction and Lemma 2.4, we can show that $s(n)$ is strictly monotone.

Lemma 2.5 *At the nonsingular point $(0,0)$, the degree n local implicit approximation $g^n(x,y)$ of the degree $m > n$ properly parameterized parametric curve $\mathbf{r}(t) = (x(t), y(t))$ is irreducible.*

Proof: Suppose $g^n(x,y)$ is reducible, and $g^n = g^k g^l$, where $n = k + l$ and $k, l > 0$. Since g^n contains linear terms, one of the g^k and g^l must have a constant term. Let $g^k(x,y) = \sum_{i+j=1}^{k} p_{ij}x^i y^j$, where p_{10} and p_{01} are not both zero, and $g^l(x,y) = \sum_{i+j=0}^{l} q_{ij}x^i y^j$, where $q_{00} \neq 0$. Let also that $g^n(x(t),y(t)) = \sum_{i=1}^{mn} \alpha_i t^i$, $g^k(x(t),y(t)) = \sum_{i=1}^{mk} \beta_i t^i$, and $g^l(x(t),y(t)) = \sum_{i=0}^{ml} \gamma_i t^i$, where $\gamma_0 = q_{00}$. We thus have

$$\sum_{i=1}^{mn} \alpha_i t^i = (\sum_{i=1}^{mk} \beta_i t^i)(\sum_{i=0}^{ml} \gamma_i t^i)$$

The coefficients of $g^n(x,y)$ are computed by solving the nonsingular system as (2.4) for some $s \geq \varphi(n)$. Moreover, $s(n)$, the order of contact of g^n, is greater than or equal to s. Thus the coefficients of g^n satisfy the linear system $\alpha = 0, \alpha_1^n = 0, \alpha_2^n = 0, \ldots, \alpha_{s(n)}^n = 0$, where, without loss of generality, we assume that $\alpha = e_{10} - 1 = 0$.

The above linear system can be represented in terms of β_i and γ_i as follows:

$$\begin{cases} q_{00}p_{10} = 1 \\ q_{00}\beta_1 = 0 \\ q_{00}\beta_2 + \beta_1\gamma_1 = 0 \\ q_{00}\beta_3 + \beta_2\gamma_1 + \beta_1\gamma_2 = 0 \\ \vdots \\ q_{00}\beta_{s(n)} + \beta_{s(n)-1}\gamma_1 + \cdots + \beta_1\gamma_{s(n)-1} = 0 \end{cases}$$

which implies $q_{00}p_{10} = 1$ and $\beta_1 = \beta_2 = \cdots = \beta_{s(n)} = 0$. Thus g^k has either order of contact larger than or equal to $s(n)$ if $s(n) < km$, or $g^k(x(t), y(t)) = 0$ for all t if $s(n) \geq km$. The first result contradicts the fact that $s(n)$ is strictly monotone, and the second contradicts the irreducibility of the exact implicitization of $\mathbf{r}(t)$. Thus g^n is irreducible. ■

When the origin is a singular curve point, the implicit approximation is not always irreducible. For example, the degree n implicit approximation of $\mathbf{c}_3(t) = (t^2, t^5)$, with implicit form $x^5 - y^2 = 0$, is $y^n = 0$. Note that $y = 0$ is the curve tangent at the origin.

2.1.3 Error Analysis

2.1.3.1 Quality of the Approximation

Given ϵ, let $T(\epsilon, n) > 0$ be such that for all $|t| < T(\epsilon, n)$ the orthogonal distance $d(t, n)$ between point $(x(t), y(t))$ and the degree n approximation $g^n(x, y) = 0$ is less than ϵ, assuming that $(x(t), y(t))$ is a regular curve point for all $|t| < T(\epsilon, n)$. The distance $d(t_p, n)$ from a point $P = (x_p, y_p) = (x(t_p), y(t_p))$ on the curve $\mathbf{r}(t)$ to the degree n approximation $g^n(x, y) = 0$ is the solution of a difficult nonlinear system. A reasonable estimate of $d(t_p, n)$ would be the distance to the $g^n(x, y) = 0$ in a direction orthogonal to the level curve $g^n(x, y) = c$, where $c = g^n(x_p, y_p)$, denoted by $d'(t_p, n)$. Note that $d'(t, n) \geq d(t, n)$ since $d(t, n)$ is the shortest distance from the point to the curve. Let $P' = (x'_p, y'_p)$ be the point on $g^n(x, y) = 0$ on which

$g^n(x,y) = 0$ intersects the line orthogonal to level curve $g^n(x,y) = c$ at P, see Figure 2.1. The value of g^n at P' expressed as its Taylor series about P is

$$g^n(x_p', y_p') = g^n(x_p, y_p) + d'(t_p, n) \cdot \nabla g^n(x_p, y_p) + \textit{higher order terms}$$

Taking the linear term, since $g^n(x_p', y_p') = 0$, $d'(t_p, n)$ can be approximated by $d''(t_p, n)$ where

$$d''(t_p, n) = \frac{g^n(x_p, y_p)}{||\nabla g^n(x_p, y_p)||} = \frac{g^n(x(t_p), y(t_p))}{[(g_x^n(x(t_p), y(t_p)))^2 + (g_y^n(x(t_p), y(t_p)))^2]^{1/2}}$$

Note that $d''(t,n)$ may be less than, greater than, or equal to $d(t,n)$ although $d'(t,n)$ is always greater or equal to $d(t,n)$.

We have found no method for computing $T(\epsilon, n)$ analytically. However, in practice we only need a method of obtaining a reasonably good estimate of $T(\epsilon, n)$. Thus it is desirable to determine $T'(\epsilon, n)$, for given ϵ and n, such that $d''(t,n) \leq \epsilon$ for $|t| \leq T'(\epsilon, n)$.

Since $2ab \leq a^2 + b^2$ for any positive a and b, we have $\sqrt{|a|\,|b|} \leq \frac{|a|+|b|}{2} \leq \sqrt{\frac{a^2+b^2}{2}}$, so that

$$d''(t,n) \leq \hat{d}(t,n) \equiv \frac{\sqrt{2} g^n(x(t), y(t))}{|g_x^n(x(t), y(t))| + |g_y^n(x(t), y(t))|}$$

When tracing $\mathbf{r}(t)$, we can detect the first value of t such that $\hat{d}(t,n) \leq \epsilon$ and $\hat{d}(t + \Delta t, n) > \epsilon$, where Δt is the step distance for t.

2.1.3.2 Curve Translation to the Origin

In the derivation of the approximant we assume that $\mathbf{r}(0) = (0,0)$, i.e. we require that $\mathbf{r}(t)$ be translated to the origin and reparameterized. Since this may incur additional inaccuracies we comment on it now.

Translation of $\mathbf{r}(t)$ to the origin is a simple operation that incurs only small errors. For, with $p = (u, v)$ as the curve point to be brought to the origin, the translated curve is simply

$$x_1(t) = x(t) - u = (p(t) - uw(t))/w(t)$$
$$y_1(t) = y(t) - v = (q(t) - vw(t))/w(t)$$

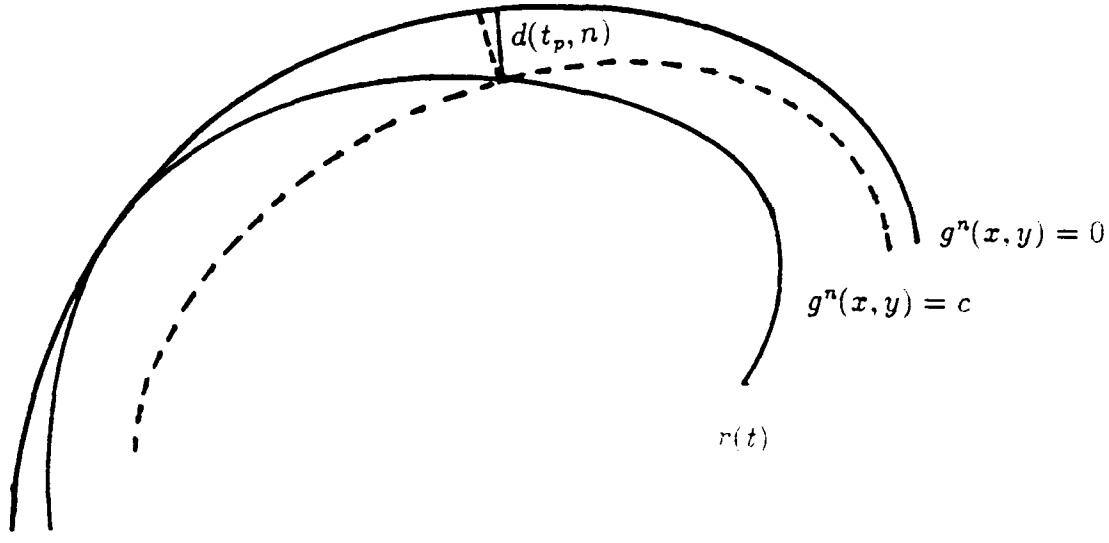


Figure 2.1 Error estimate

So, we have to subtract two polynomials in order to bring p to the origin.

Now assume that $\mathbf{r}(t_0) = p$, and consider reparameterization such that p not only is moved to the origin but that also $t_0 = 0$, for the reparameterized curve. Here we need to substitute $\bar{t} + t_0$ for t, i.e.

$$x_2(\bar{t}) = x_1(\bar{t} + t_0)$$
$$y_2(\bar{t}) = y_1(\bar{t} + t_0)$$

is the final curve. As observed in the introduction, although substitution is conceptually simple, it nonetheless may introduce numerical errors that could be significant. According to experiments by Prakash and Patrikalakis [54], Kahan's method described in [39] exhibits good numerical stability, and offers one method of implementing the needed reparameterization.

A second method would be to avoid reparameterization altogether by reformulating the derivation of the approximant given before. That is, we consider $\mathbf{r}(t)$ containing the origin at which t is not necessarily 0, seeking again an implicit approximant at the origin. Clearly this is possible and requires only straightforward

modifications of our method. In fact, even translation of the point to the origin can be avoided by such modifications. The details are routine.

2.1.4 Experiments

A good local approximation of a curve will provide a more accurate local approximation and larger interval $T(\epsilon, n)$ of approximation. for a given ϵ, when the degree n of the approximation increases. The local implicit approximation $g^n(x, y) = 0$ of $\mathbf{r}(t)$ is determined by $\varphi(n)$ linear conditions imposed on its coefficients, where $\varphi(n)$ is the degree of freedom of $g^n(x, y)$. Thus, as n increases, more conditions can be satisfied and finally the exact implicitization is obtained when $n = m$. Hence our local implicit approximation is capable of approximating a given curve not only locally but also globally in the sense that $T(\epsilon, n)$, for a given ϵ, will be larger when n increases. On the other hand, a local explicit approximation is limited due to the asymmetry introduced by making one variable an explicit function of the other. Thus, an local explicit approximation can only approximates the given curve locally for $|x| < R$, where R is its radius of convergence, no matter how high the degree of approximant is.

We give as example the approximation of several parametric curves that are not singular at the origin, showing both implicit and explicit approximations.

Example 2.4 Four curve examples are shown below.

$\mathbf{c}_4(t) = (t^6 + t^5 - 2t^3 + 3t^2 + 12t, t^6 - t^5 + t^4 - 4t^3 - 2t^2 + 24t)$

$\mathbf{c}_5(t) = (3t^6 - 4t^5 - 8t^3 + 6t^2 + 3t, -3t^6 + 4t^5 + 5t^4 - 6t^3 - 8t^2 + 3t)$

$\mathbf{c}_6(t) = (3t^6 + t^5 - 2t^4 + 38t^3 - 5t^2 - 14t, t^6 - 12t^5 - 2t^4 + 2t^3 - 7t^2 + 13t)$

$\mathbf{c}_7(t) = ((t^6+3t^5-6t^4+4t^3-36t^2+36t)/w(t), (3t^6+t^5-2t^4+39t^3-69t^2+33t)/w(t))$

where $w(t) = 7t^6 + 10t^5 + 9t^4 + 6t^2 + 3t + 7$.

The curves of $\mathbf{c}_4(t)$, $\mathbf{c}_5(t)$, $\mathbf{c}_6(t)$ and $\mathbf{c}_7(t)$ with t in $[-1, 1]$, and their local implicit approximations and local explicit approximation are shown in Figures 2.2, 2.3, 2.4 and 2.5. Note the good quality of local implicit approximation.

Tables 2.1 2.2 and 2.3, for $\mathbf{c}_4(t)$,$\mathbf{c}_5(t)$ and $\mathbf{c}_6(t)$ respectively, list the y-values of a sequence of x-values to quantify how accurately the low degree local implicit forms approximate the original curves. The corresponding values of local explicit forms are also listed for comparison. For such examples, we observe that

(a) For local implicit approximation, $d(t, n+k) < d(t, n)$ for t in $[-1, 1]$ and $k \geq 1$. In addition, $T(\epsilon, n) < T(\epsilon, n+k)$ for $k \geq 1$.

(b) $T(\epsilon, 2)$ of local implicit approximation is greater than $T(\epsilon, 6)$ of local explicit approximation.

(c) Degree 2 and 3 local implicit approximations give very accurate approximations on a reasonable range of t.

(d) Degree 5 local implicit approximation approximates the original curve very precisely at least for $-1 \leq t \leq 1$. ■

When computing an explicit approximation $y = h(x)$ directly from the curve $\mathbf{r}(t)$, we first compute the degree n power series $t = \sum_{i=1}^{n} d_i x^i$ from $x = x(t)$, and then substitute it for t in $y = y(t)$. As a result, only the first n coefficients of $h(x)$ are exact and the remaining coefficients obtained in the computation should be discarded. Moreover, substituting $t = \sum_{i=1}^{n} d_i x^i$ for t in $y = y(t)$ is not a cheap computation, especially for high degree local explicit approximations. Hence, the computation of a local explicit approximations of a parametric curve directly from $\mathbf{r}(t)$ is more costly than the implicit form. In general, the computation of local implicit approximation involves generating the α_i^n and solving the linear system, which is fairly efficient for low-degree approximation.

The local explicit approximation is an analytic function that does not exist at a curve singularity. In contrast, a local implicit approximation always exists.

Example 2.5 Local implicit approximations can be derived at singularities, including cusps, where local explicit approximation fails. Let $\mathbf{c}_8(t) = (5t^3 + 2t^2, t^4 - 3t^3 + 2t^2)$ with the implicit form $f^4(x, y) = -x^4 + 55x^3 + 683x^2y + 1325xy^2 + 625y^3 - 336x^2 + 672xy - 336y^2$. The origin is a cusp of $\mathbf{c}_8(t)$ with tangent $x - y = 0$. The

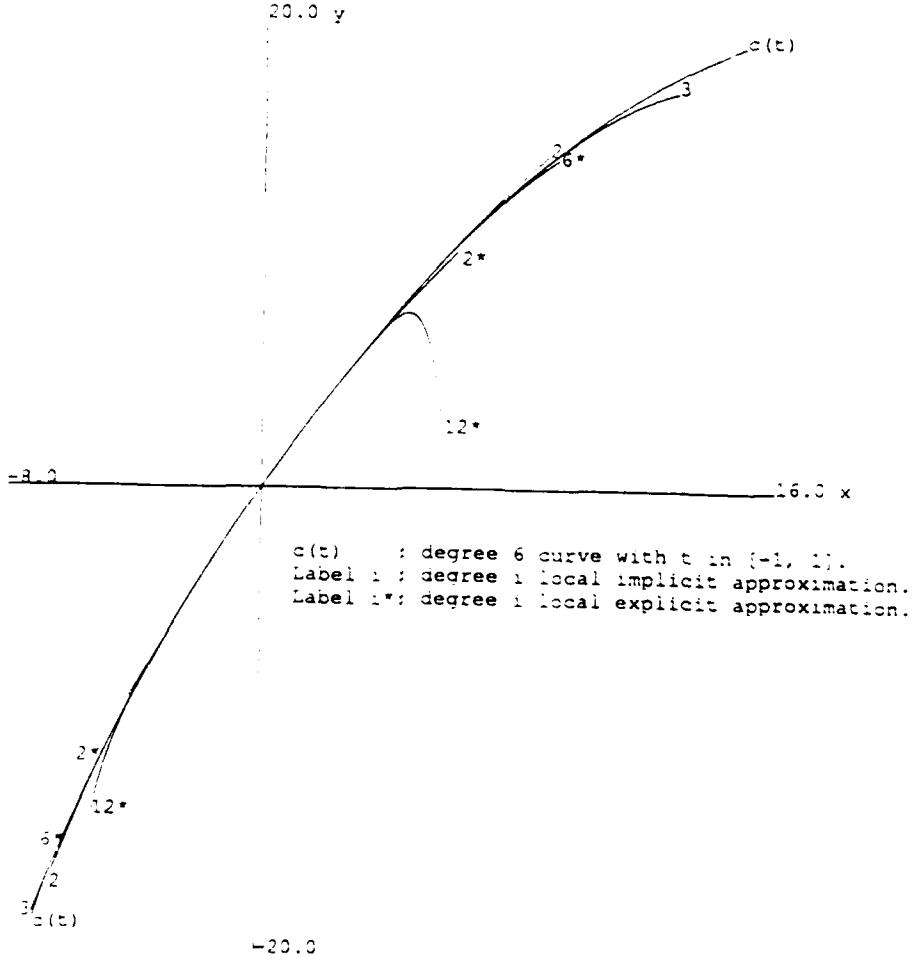


Figure 2.2 Curve $\mathbf{c}_1(t)$

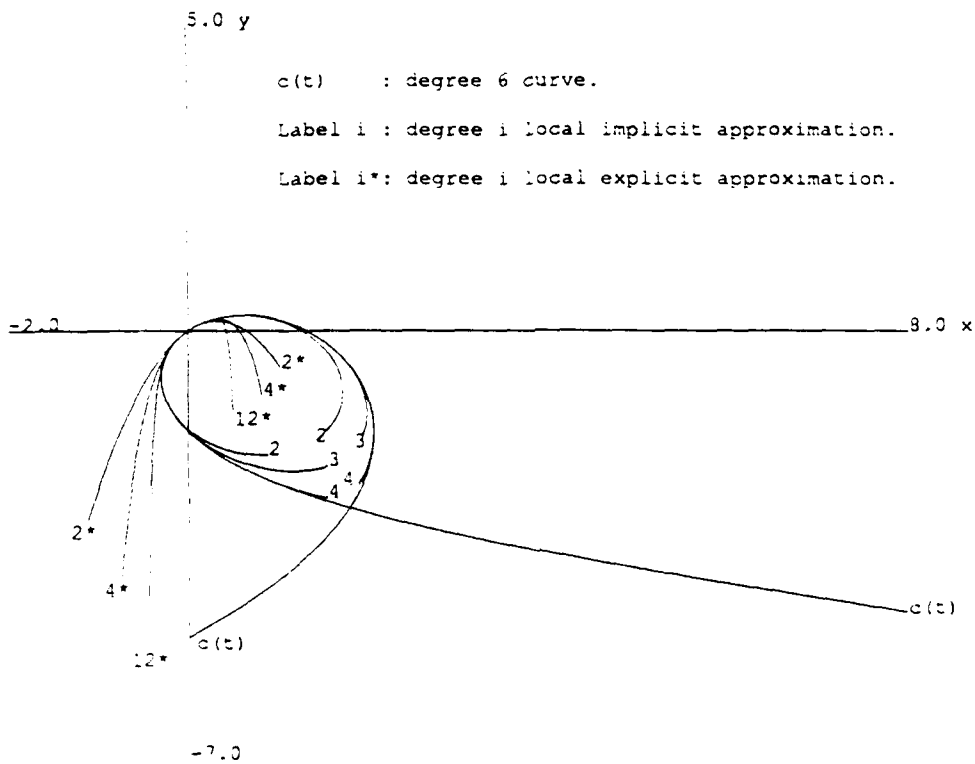


Figure 2.3 Curve $\mathbf{c}_5(t)$

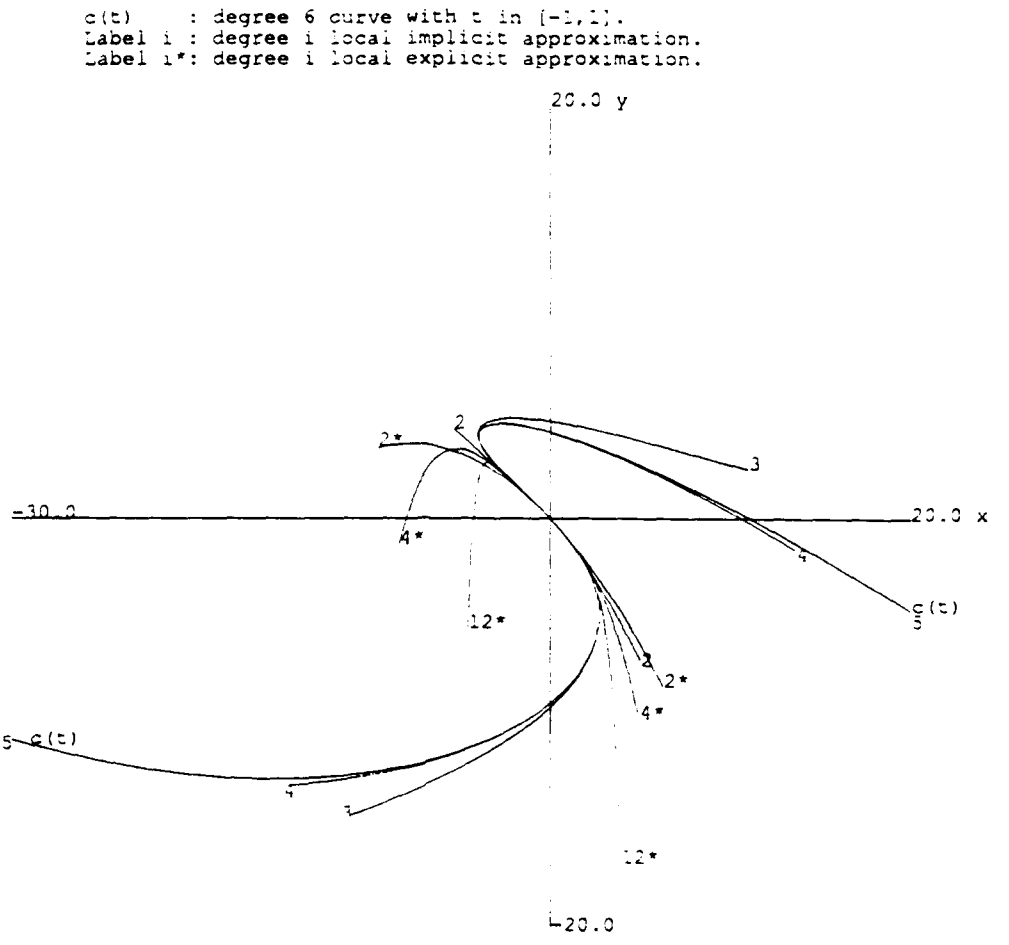


Figure 2.4 Curve $\mathbf{c}_6(t)$

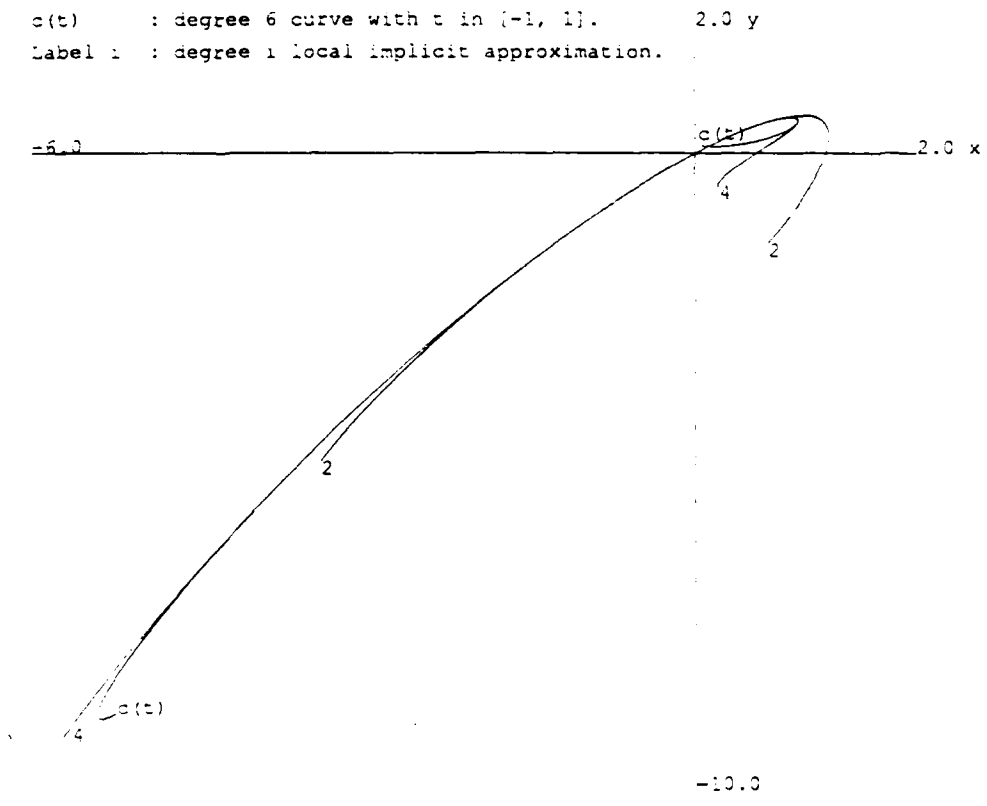


Figure 2.5 Curve $\mathbf{c}_7(t)$

Table 2.1 Error of approximation for curve $\mathbf{c}_4(t)$

li n : local implicit form of degree n. le n : local explicit form of degree n.

	x value								
	-5.0	-3.0	-2.0	-1.25	0	1.25	2.0	4.0	7.0
exact	-11.845579	-6.5811195	-4.2442303	-2.5918415	0	2.4172442	3.793271	7.213065	11.663555
li 4	-11.845584	-6.5811195	-4.2442300	-2.5918415	0	2.4172442	3.793271	7.213066	11.663522
li 3	-11.843186	-6.5811080	-4.2442300	-2.5918415	0	2.4172442	3.793271	7.213022	11.658528
li 2	-11.910466	-6.5846540	-4.2446346	-2.5918765	0	2.4172716	3.793540	7.220462	11.761091
le 2	-23.889015	-11.000046	-6.2222424	-3.3680634	0	1.6319366	1.777758	-0.8889694	-13.222469
le 4	-24.304024	-11.078828	-6.2439766	-3.3730750	0	1.6359663	1.793057	-0.79231834	-12.91141
le 6	-24.336723	-11.081027	-6.2442436	-3.3730989	0	1.6359847	1.793233	-0.7881178	-12.878107
le 12	-25.297968	-11.082955	-6.2442613	-3.3730989	0	1.6359840	1.793167	-0.9260575	-103.35114

Table 2.2 Error of approximation for curve $c_5(t)$

li n : local implicit form of degree n. le n : local explicit form of degree n.

	x value							
	-0.25	-0.1	0.0	0.1	0.25	1.0	1.25	1.5
exact	-0.4291671 -1.1245294	-0.11828928 -1.4897133	0 -1.6495429	0.08622920 -1.7816917	0.17517897 -1.9479930	0.17769729 -2.5130520	0.043922530 -2.6506336	-0.16892694 -2.7737687
li 4	-0.4291671 -1.1245286	-0.11828928 -1.4897133	0 -1.6495163	0.08622920 -1.7816285	0.17517897 -1.9478126	0.17769726 -2.5069103	0.043921884 -2.6342275	-0.168934 -2.7259927
li 3	-0.4291676 -1.1233613	-0.11828928 -1.4840809	0 -1.6388266	0.08622920 -1.7638947	0.17517897 -1.9152321	0.17769955 -2.2900155	0.043849964 -2.2978785	-0.16974114 -2.2328222
li 2	-0.4286827 -1.1285945	-0.11828820 -1.4734732	0 -1.6147509	0.08622874 -1.7239690	0.17514941 -1.8473737	0.16556032 -2.0102053	0.008065359 -1.91018380	-0.27427490 -1.685317
le 12	-0.42683282 ——-	-0.11828927 ——-	0 ——-	0.0862292 ——-	0.17486227 ——-	-9337.529 ——-	-143129.16 ——-	-1323236.8 ——-
le 6	-0.41469532 ——-	-0.11827987 ——-	0 ——-	0.0862238 ——-	0.17268895 ——-	-18.46853 ——-	-75.29226 ——-	-234.37852 ——-
le 4	-0.39822853 ——-	-0.11815020 ——-	0 ——-	0.0861461 ——-	0.16890110 ——-	-2.872428 ——-	-7.885883 ——-	-17.354166 ——-
le 2	-0.34722220 ——-	-0.11555555 ——-	0 ——-	0.0844444 ——-	0.15277778 ——-	0.55555556 ——-	-1.1805556 ——-	-2.0 ——-

Table 2.3 Error of approximation for curve $\mathbf{c}_6(t)$

li n : local implicit form of degree n. le n : local explicit form of degree n.

	x value								
	-3.5	-2.5	-0.5	-0.25	0	0.25	0.5	2.0	2.5
exact	3.1174486	2.1534863	.45135155	.22867906	0	-.23612773	-.48143223	-2.3418179	-3.4202664
li 4	3.1174495	2.1534863	.45135155	.22867906	0	-.23612773	-.48143226.	-2.3418179	-3.4202664
li 3	3.1159813	2.1534630	.45135158	.22867908	0	-.23612775	-.48143230	-2.3418021	-3.4196750
li 2	2.9099880	2.1148510	.45132570	.22867821	0	-.23612662	-.48139048	-2.2394760	-2.0944711
le 12	2.7423260	2.1480120	.45135152	.22867905	0	-.23612773	-.48143230	-2.3401886	-3.3552332
le 6	2.8504105	2.1270490	.45135110	.22867908	0	-.23612775	-.48143163	-2.3187237	-3.2120173
le 4	2.7878397	2.0934718	.45132630	.22867824	0	-.23612681	-.48139983	-2.2753644	-3.0687234
le 2	2.5223215	1.9501640	.44943514	.22843021	0	-.23585550	-.47913630	-2.0947520	-2.6926932

degree 2 local implicit approximation is a double line $(x-y)^2$, which is the best degree 2 approximation one can derive at cusp. The degree 3 local implicit approximation is $x^2-2xy+y^2-0.16259766x^3-2.0356445x^2y-3.940918xy^2-1.8608398y^3$ which shows very nice approximation to the $\mathbf{c}_8(t)$ with t in $[-1,1]$, see Figure 2.6. As a next example, we consider $\mathbf{c}_9(t) = ((5t^5-16t^4+10t^3+4t^2)/w(t), (t^5+t^4+2t^3-16t^2)/w(t))$, where $w(t) = 0.1t^3+0.1t^2-2t+12.5$. The $\mathbf{c}_9(t)$ is a singular curve with a cusp at the origin and a self-intersection as well, as shown in Figure 2.7. Figure 2.8 shows the degree 3 and degree 4 local implicit approximations of $\mathbf{c}_9(t)$. The degree 4 local implicit approximation shows remarkable performance. ■

2.2 Local Implicit Approximation of Parametric Surfaces

We derive an implicit surface $g(x,y,z)=0$ that approximates the parametric surface $\mathbf{P}(s,t) = (x(s,t), y(s,t), z(s,t))$ at the origin to a specified order of contact, using the method of Section 2.1. Let

$$\mathbf{P}(s,t) = (x(s,t), y(s,t), z(s,t)) = (\frac{p(s,t)}{w(s,t)}, \frac{q(s,t)}{w(s,t)}, \frac{r(s,t)}{w(s,t)})$$

be a rational parametric surface of total degree m containing the origin, where

$$p(s,t) = \sum_{i+j=1}^{m} a_{ij}s^it^j, \qquad q(s,t) = \sum_{i+j=1}^{m} b_{ij}s^it^j,$$

$$r(s,t) = \sum_{i+j=1}^{m} c_{ij}s^it^j, \qquad w(s,t) = \sum_{i+j=0}^{m} d_{ij}s^it^j$$

with a_{ij}, b_{ij}, c_{ij}, $i+j=m$, not all zeros and $d_{00} \neq 0$. It has been shown by Macaulay [46] that a parametric surface of the above form has an irreducible implicit form $f^d(x,y,z)=0$ of degree $d \leq m^2$.

Let $g^n(x,y,z) = \sum_{i+j+k=1}^{n} e_{ijk}x^iy^jz^k$, $n \leq m^2$, with symbolic coefficients e_{ijk} be a general implicit form of degree n surface that contains the origin. Since $g^n(x,y,z)=0$ is unique up to a constant factor, it has degrees of freedom $\rho(n) = ((n+1)(n+2)(n+3))/6 - 2$.

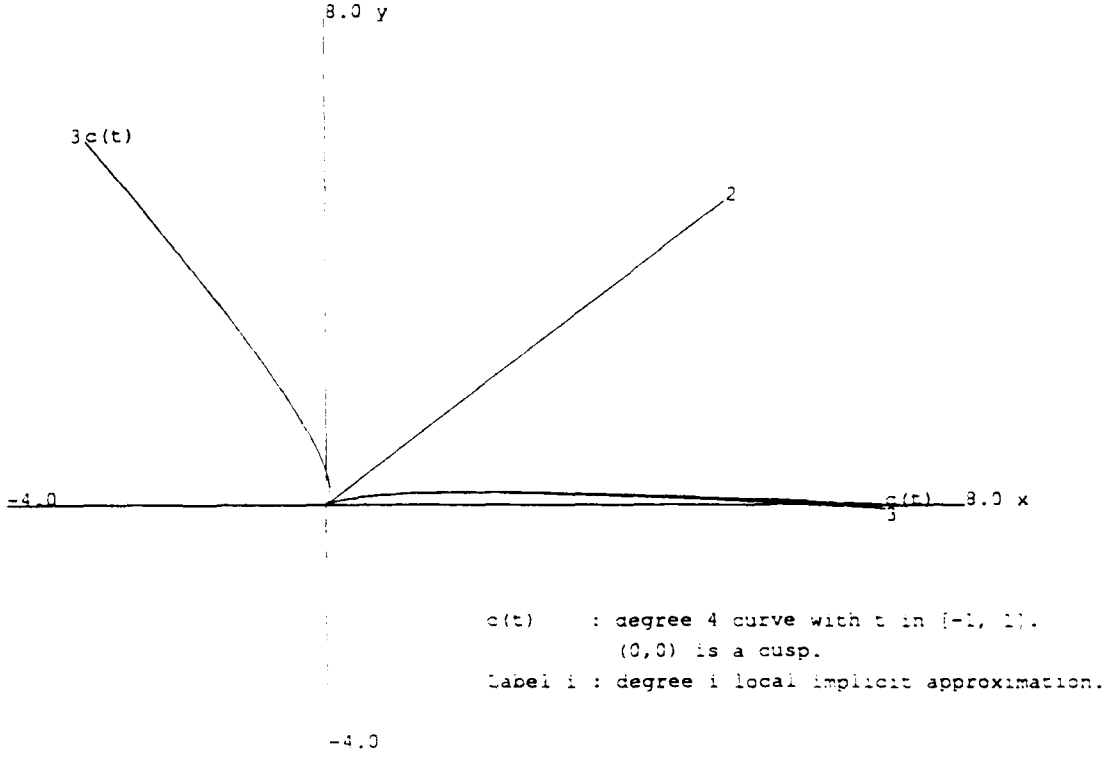


Figure 2.6 Curve $\mathbf{c}_8(t)$

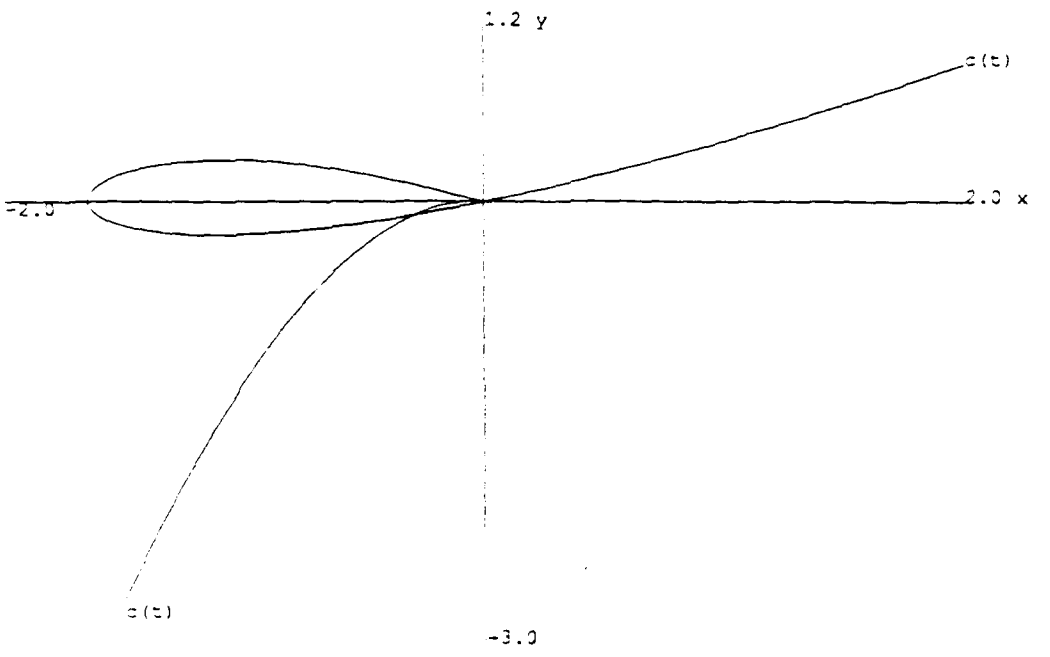


Figure 2.7 Curve $c_9(t)$

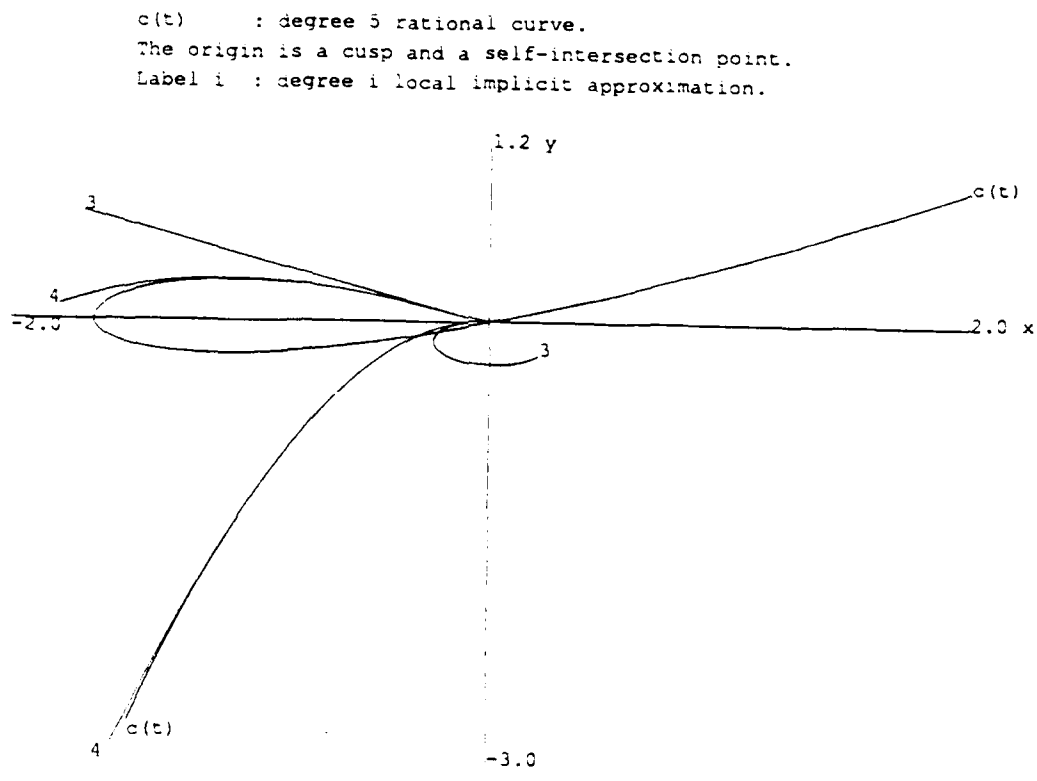


Figure 2.8 Curve $\mathbf{c}_9(t)$

When substituting $x(s,t), y(s,t)$ and $z(s,t)$ into $g^n(x,y,z)$, we obtain

$$g^n(x(s,t),y(s,t),z(s,t)) = \frac{G^n(p(s,t),q(s,t),r(s,t),w(s,t))}{(w(s,t))^n} = \frac{\sum_{i+j=1}^{nm} \alpha_{ij} s^i t^j}{(w(s,t))^n}$$

where $G^n(x,y,z,w)$ is the homogeneous form of $g^n(x,y,z)$, and the α_{ij} are linear combinations of e_{ijk}. The local implicit approximation $g^n(x,y,z)$ of the parametric surface $\mathbf{P}(s,t)$ is computed as in the case of the curve approximation. The following section shows a recursive derivation of α_{ij} that obviates the need to explicitly substitute.

2.2.1 A Recurrence for α_{ij}

Let $G^n(x,y,z,w)$ and $G^{n-1}(x,y,z,w)$ denote the homogeneous polynomials of $g^n(x,y,z)$ and $g^{n-1}(x,y,z)$, respectively. Since

$$g^n(x,y,z) = g^{n-1}(x,y,z) + \sum_{i+j+k=n} e_{ijk} x^i y^j z^k$$

we have

$$G^n(x,y,z,w) = wG^{n-1}(x,y,z,w) + \sum_{i+j+k=n} e_{ijk} x^i y^j z^k \tag{2.5}$$

We define $(a(k))_{ij}$, $(b(k))_{ij}$ and $(c(k))_{ij}$ by setting

$$(p(s,t))^k = (\sum_{i+j=1}^{m} a_{ij} s^i t^j)^k = \sum_{i+j=k}^{km} (a(k))_{ij} s^i t^j$$

and similarly for

$$(q(s,t))^k = \sum_{i+j=k}^{km} (b(k))_{ij} s^i t^j$$

and

$$(r(s,t))^k = \sum_{i+j=k}^{km} (c(k))_{ij} s^i t^j$$

Since

$$\sum_{i+j=k}^{km} (a(k))_{ij} s^i t^j = \left(\sum_{p+q=k-1}^{(k-1)m} (a(k-1))_{pq} s^p t^q \right) p(s,t)$$

as shown in [48], the $(a(k))_{ij}$ is recursively defined as

$$(a(k))_{ij} = \begin{cases} 1 & \text{if } k = i = j = 0 \\ 0 & \text{if } k = 0 \text{ and } i + j \geq 1 \\ a_{ij} & \text{if } k = 1 \text{ and } i + j \geq 1 \\ \sum_{k-1 \leq p+q \leq i+j-1} (a(k-1))_{pq} a_{(i-p)(j-q)} & \text{if } k \geq 2 \text{ and } i + j \geq k \end{cases}$$

$(b(k))_{ij}$ and $(c(k))_{ij}$ are defined analogously.

Recursive derivations for $(a(k))_{ij}$, $(b(k))_{ij}$ and $(c(k))_{ij}$ can be found in [48]. Let α_{ij}^n and α_{ij}^{n-1} be the coefficient of $s^i t^j$ in $G^n(p(s,t), q(s,t), r(s,t), w(s,t))$ and $G^{n-1}(p(s,t), q(s,t), r(s,t), w(s,t))$, respectively. From (2.5), α_{ij}^n can be derived from the α_{kl}^{n-1}, where $1 \leq k \leq i$ and $1 \leq l \leq j$, as shown in the following formula:

$$\begin{aligned} \alpha_{ij}^n &= \textit{coefficient of } s^i t^j \textit{ in } (w(s,t) \sum_{i+j=1}^{m(n-1)} \alpha_{ij}^{n-1} s^i t^j \\ &\quad + \sum_{k_1+k_2+k_3=n} e_{k_1 k_2 k_3} (p(s,t))^{k_1} (q(s,t))^{k_2} (r(s,t))^{k_3}) \\ &= \sum_{l_1=1}^{i} \sum_{l_2=1}^{j} \alpha_{l_1 l_2}^{n-1} d_{(i-l_1)(j-l_2)} \\ &\quad + \sum_{k_1+k_2+k_3=n} \sum_{\substack{p_1+p_2+p_3=i \\ q_1+q_2+q_3=j}} e_{k_1 k_2 k_3} (a(k_1))_{p_1 q_1} (b(k_2))_{p_2 q_2} (c(k_3))_{p_3 q_3} \end{aligned}$$

Note that $\alpha_{ij}^1 = e_{100} a_{ij} + e_{010} b_{ij} + e_{001} c_{ij}$.

For an integral parametric surface $\mathbf{P}(s,t)$, since $(a(k))_{ij} = 0, (b(k))_{ij} = 0$ and $(c(k))_{ij} = 0$ for $i + j < k$ and $\alpha_{ij}^n = 0$ for $i + j > (n-1)m$, we have

for $1 \leq i + j \leq n - 1$,

$$\alpha_{ij}^n = \alpha_{ij}^{n-1}$$

for $n \leq i + j \leq (n-1)m$,

$$\alpha_{ij}^n = \alpha_{ij}^{n-1} + \sum_{k_1+k_2+k_3=n} \sum_{\substack{p_1+p_2+p_3=i \\ q_1+q_2+q_3=j}} e_{k_1 k_2 k_3} (a(k_1))_{p_1 q_1} (b(k_2))_{p_2 q_2} (c(k_3))_{p_3 q_3}$$

for $(n-1)m < i+j \leq nm$,

$$\alpha_{ij}^n = \sum_{k_1+k_2+k_3=n} \sum_{\substack{p_1+p_2+p_3=i \\ q_1+q_2+q_3=j}} e_{k_1k_2k_3}(a(k_1))_{p_1q_1}(b(k_2))_{p_2q_2}(c(k_3))_{p_3q_3}$$

2.2.2 Derivation of the Method

2.2.2.1 Rank of the Linear System

Having derived α_{ij}^n, $i+j = 1, 2, \ldots, nm$, for the degree n implicit approximation $g^n(x, y, z)$, we write the system of linear equations $\alpha_{10}^n = 0$, $\alpha_{01}^n = 0$, $\alpha_{20}^n = 0$, $\alpha_{11}^n = 0$, $\alpha_{02}^n = 0$, $\ldots$, $\alpha_{(nm)0}^n = 0$, $\alpha_{(nm-1)1}^n = 0$, $\ldots$, $\alpha_{1(nm-1)}^n = 0$, $\alpha_{0(nm)}^n = 0$ in matrix form

$$\mathbf{A}_{mn}\mathbf{e}_n = \mathbf{0} \tag{2.6}$$

where $\mathbf{e}_n = (e_{100}, e_{010}, e_{001}, e_{200}, e_{110}, e_{101}, e_{020}, e_{011}, e_{002}, \ldots, e_{00n})^T$ is the vector of unknowns. $\mathbf{A}_{mn}$ so defined is of dimension $((nm+1)(nm+2))/2-1$ by $\rho(n)+1$, and has rank at most $\rho(n)+1$. As in the curve case, the rank of $\mathbf{A}_{mn}$ is critical when solving for the unknown coefficients e_{ijk}. The following theorem characterizes the rank of $\mathbf{A}_{mn}$.

Theorem 2.2 *If $\mathbf{P}(s,t)$ is a properly parameterized rational surface of total degree m, then*

$$rank(\mathbf{A}_{mn}) = \begin{cases} \rho(n)+1 & \text{if } n < d \\ \rho(n) & \text{if } n = d \end{cases}$$

where d ($\leq m^2$) is the degree of the implicit form of $\mathbf{P}(s,t)$, n is the degree of the approximant, and $\rho(n)$ is the degree of freedom of the approximant.

Proof: Similar to proofs of Lemma 2.2 and Lemma 2.3. ■

As a result of Theorem 2.2, it is clear that the exact implicitization of $P(s,t)$ is the solution of $\mathbf{A}_{mm}\mathbf{e}_m = \mathbf{0}$, with one variable set to a fixed value.

2.2.2.2 The Algorithm

We compute the degree n implicit approximation $g^n(x,y,z)$ as follows: Let

$$\mathbf{B}\mathbf{e}_n = \mathbf{0} \tag{2.7}$$

be the subsystem of (2.6) that consists of the first s equation of (2.6) such that $\mathbf{B}$ has rank $\rho(n)$. Augment the system (2.7) with $\alpha = 0$, where α is determined as follows: if the origin is a regular surface point, α is $e_{100} - 1$, $e_{010} - 1$, or $e_{001} - 1$ depending on the gradient of the surface at the origin. If the origin is a singular surface point, then $\alpha = e_{ijk} - 1$ where the indices i, j, and k are selected by inspection. Thus, a linear system

$$\mathbf{C}\mathbf{e}_n = \mathbf{b} \tag{2.8}$$

is obtained. System (2.8) may be an inconsistent system. If this happens, some equations must be removed from (2.8) to ensure the consistency.

One alternative for handling inconsistencies is that we replace $\alpha = 0$ in (2.8) with $e_{n00} - 1 = 0$, $e_{0n0} - 1 = 0$, or $e_{00n} - 1 = 0$ and then solve it as usual. Examples show that $g^n(x,y,z)$ computed by this method can be of the form $(ax+by+cz)^n$ for some a, b, c, that is, it degenerates to the tangent plane. To remove this degeneracy, we do the following:

1. Solve for $g^1(x,y,z)$, and compute

$$\sum_{i+j=n} \beta_{ij} s^i t^j = (g^1(x(s,t), y(s,t), z(s,t)))^n$$

2. Consider the linear system that consists of the first s' equations of (2.6) and $\alpha = 0$, where $\alpha = 0$ is $e_{n00} - 1 = 0$, $e_{0n0} - 1 = 0$, or $e_{00n} - 1 = 0$ and s' is chosen such that the coefficient matrix of the system has rank $\rho(n)$.

3. Find a β_{ij} which is nonzero and augment the corresponding $\alpha_{ij} = 0$ to the above system, then solve it.

This computation of the local implicit approximation results in an approximant that has roughly $n^{3/2}$-th order of contact. Thus, when raising the degree of the approximant, the order of contact with the surface $\mathbf{P}(s,t)$ grows subquadratically.

Example 2.6 Consider

$$\mathbf{P}(s,t) = (x(s,t)/w(s,t), y(s,t)/w(s,t), z(s,t)/w(s,t))$$

where

$$\begin{aligned} x(s,t) &= -200t^2 + 12st + 400t - 200s^2 - 10s \\ y(s,t) &= 15t^2 - 14st + 10t - 11s^2 + 400s \\ z(s,t) &= 200t^2 + 11st - t + 200s^2 + 2s \\ w(s,t) &= 100t^2 - 200t + 100s^2 + 200 \end{aligned}$$

We compute degree 2 and degree 3 local implicit approximations

$$\begin{aligned} &g^2(x,y,z) = \\ &\quad -108.44294z^2 - 10.264638yz - 13.162097xz + 381.19047z \\ &\quad - 95.092836y^2 - 5.241114xy - 1.8809524y - 94.85476x^2 + x \end{aligned}$$

and

$$\begin{aligned} &g^3(x,y,z) = \\ &\quad 1.3012126z^3 + 5.16125yz^2 - 46.69081xz^2 - 103.81S164z^2 \\ &\quad - 1.1158845y^2z + 15.622598xyz + 4.518084yz \\ &\quad + 1.267575x^2z + 180.40466xz + 381.19047z - 3.6884814y^3 \\ &\quad - 48.00386xy^2 - 95.16589y^2 + 5.5613696x^2y - 6.1573525xy \\ &\quad - 1.8809524y - 44.977395x^3 - 94.34699x^2 + x \end{aligned}$$

Note that the normal of $f^4(x,y,z)$, the exact implicit form of $\mathbf{P}(s,t)$, at the origin is almost parallel to z-axis. Thus, to show the performance of the local

Table 2.4 Maximal deviations between the intersection curves

li n : local implicit form of degree n. le n : local explicit form of degree n.

degree	r=0.25	r=0.50	r=0.75	r=1.25	r=1.50	r=1.75
li 3	0.000000	0.000004	0.000035	0.000837	0.004964	0.019612
li 2	0.000289	0.002531	0.009630	0.066627	——	——

implicit approximation, we intersect the cylinder $h(x,y,z) = x^2 + y^2 - r^2 = 0$ with the surfaces f^4, g^2, and g^3, and plot the intersection curves of $f^4 = 0 \cap h = 0, g^3 = 0 \cap h = 0$, and $g^2 = 0 \cap h = 0$ in one figure. Figures 2.9 and 2.10, show the intersection curves in cylindrical coordinates, for $r = 0.25, 0.5, 0.75, 1.00, 1.25, 1.50$, and 1.75. Table 2.4 lists the maximal deviations between the intersection curves $f^4 = 0 \cap h = 0$ and $g^3 = 0 \cap h = 0$, and $f^4 = 0 \cap h = 0$ and $g^2 = 0 \cap h = 0$. ■

2.3 Remarks on Resultants

Different resultants are formulated in the classical literature for the purpose of eliminating variables from systems of algebraic equations. Early expositions of several formulations are found in Netto's book cited in the references. In essence, resultants constitute a projection. A well-known problem of elimination based on resultants is the possibility of obtaining extraneous factors. For example, when implicitizing the parametric sphere

$$\begin{aligned} x(s,t) &= (1 - s^2 - t^2)/(1 + s^2 + t^2) \\ y(s,t) &= 2t/(1 + s^2 + t^2) \\ z(s,t) &= 2s/(1 + s^2 + t^2) \end{aligned}$$

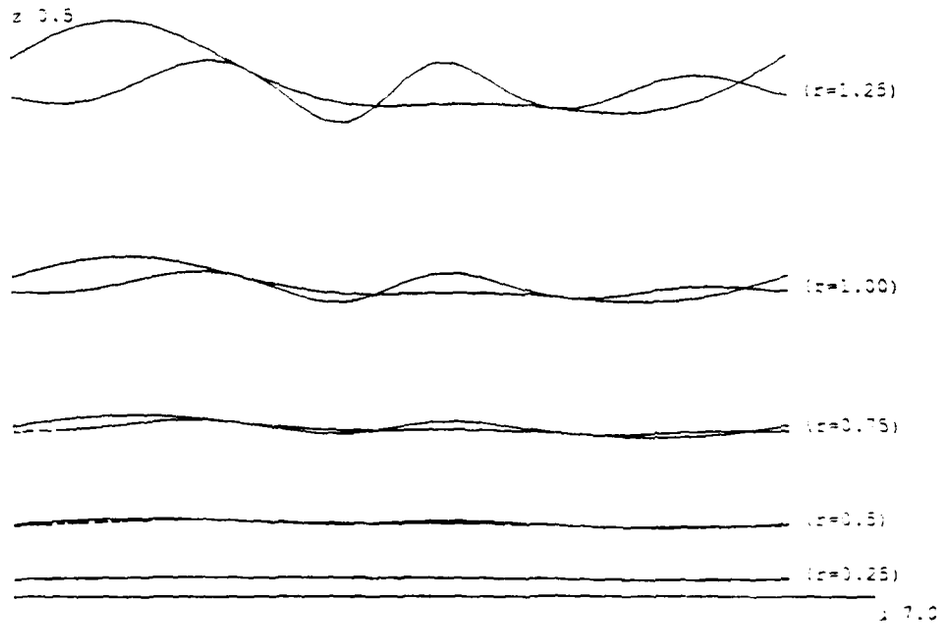


Figure 2.9 $f^4 = 0 \cap h = 0$ and $g^2 = 0 \cap h = 0$

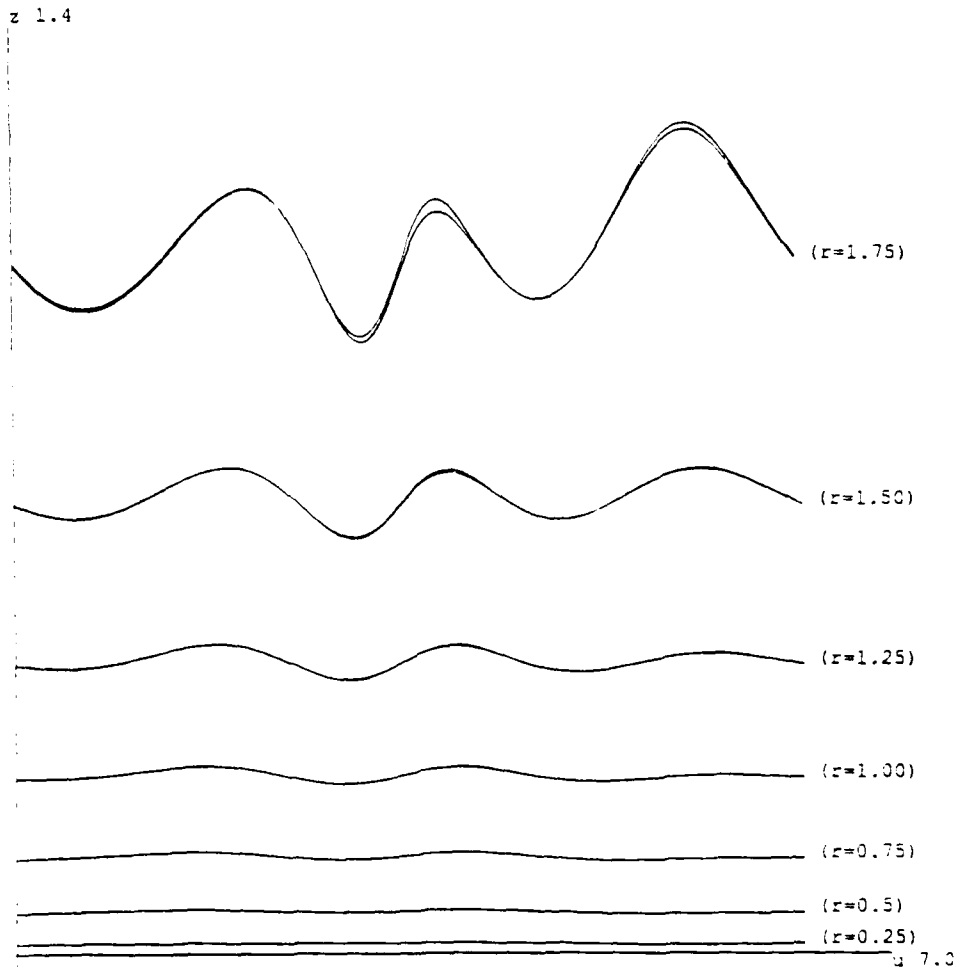


Figure 2.10 $f^4 = 0 \cap h = 0$ and $g^3 = 0 \cap h = 0$

the Sylvester resultant yields

$$256(x+1)^4(x^2+y^2+z^2-1)^2$$

and the Dixon's resultant yields

$$-64(x+1)(x^2+y^2+z^2-1)$$

In each case, an extraneous factor $x+1$, to some power, is present.

Technically, a resultant is based on formulating a system of linear equations with symbolic coefficients. This is especially apparent in the derivation of Sylvesters resultant. Macaulay recognized that extraneous factors technically are related to dependent equations, and that they can be eliminated by division by a suitable minor [47]. Modern work on the multivariate resultant tries to find this minor algorithmically, i.e., to recognize and eliminate extraneous factors. See, e.g., [20, 13]. In our approach, a linear system is formulated numerically, hence dependencies among the equations are easy to recognize. If the approximant is formulated with the exact degree of the implicit form, then our approach determines the implicit form without extraneous factors. If an approximant of higher degree is determined with our approach, then a reducible implicit form with extraneous factors could be generated.

3. LOCAL APPROXIMATIONS OF 2-D SURFACES

In solid modeling with curved surfaces, a number of desirable surface operations, including offsetting, spherical blending and the formation of Voronoi surfaces, raise difficult mathematical problems that must be solved in order to represent and interrogate the resulting surfaces. Such surfaces cannot be easily defined in the conventional 3-D space. In contrast, they can be formulated mathematically, with the theory of envelopes, in higher dimensional space in a straightforward manner. For example, given the surface $h(x, y, z) = 0$, we formulate the r-offset of h as the envelope of a family of spheres with radius r whose centers lie on the surface $h = 0$. This formulation results in a system of four polynomial equations in 6-D space. Such surfaces are generally 2-surfaces in $\mathbf{R}^n$, $n > 3$, and are defined by

$$\begin{aligned} f_1(x_1, x_2, \ldots, x_n) &= 0 \\ f_2(x_1, x_2, \ldots, x_n) &= 0 \\ &\vdots \\ f_{n-2}(x_1, x_2, \ldots, x_n) &= 0 \end{aligned} \tag{3.1}$$

where the f_i are polynomials or in matrix form as $\mathbf{F}(\mathbf{x}) = \mathbf{0}$. Although the exact closed-form representation of such a surface

$$f(x_1, x_2, x_3) = 0 \tag{3.2}$$

could be derived in principle by elimination methods such as Gröbner basis or resultant techniques, it is often not feasible to do so in practice due to the high complexity of these methods.

For surface representations in high-dimensional space to be practical, good algorithms for some basic operations have to be developed. Among them, as

mentioned in [32], are locating points on surface intersections, tracing surface intersections, and local surface approximations. It is our objective here, for a given surface representation $\mathbf{F}(\mathbf{x}) = \mathbf{0}$ and a regular point $\mathbf{x}^0 = (x_1^0, x_2^0, \ldots, x_n^0)$ on it, to develop computational schemes for the local geometry of $f(x_1, x_2, x_3) = 0$ and to investigate approximation schemes which derive a surface of lower degree that approximates the given surface locally at $\hat{\mathbf{x}}^0 = (x_1^0, x_2^0, x_3^0)$ in (x_1, x_2, x_3)-space. These techniques will be of practical interest if they are efficient and can be incorporated into algorithms for surface interrogation such as computing surface intersections.

In this chapter, we proceed as follows: Section 1 presents techniques that describe the local geometry of $f(x_1, x_2, x_3) = 0$. Section 2 presents an algorithm that computes degree two implicit approximants of $f(x_1, x_2, x_3) = 0$ without actually computing f. Section 3 describes the computations for local explicit approximants to $f(x_1, x_2, x_3) = 0$. In section 4, we derive a procedure for computing parametric approximants.

3.1 Local Geometry of the Projected Surface

For a given 2-surface S_F in $\mathbf{R}^n$ and a regular surface point $\mathbf{x}^0$, we first derive some schemes that determine the normal vector and tangent vectors of S_f at a regular projected point $\hat{\mathbf{x}}^0$ from the normals and tangents of S_F at $\mathbf{x}^0$. We then derive an algorithm to compute the normal curvature of surface S_f at $\hat{\mathbf{x}}^0$ in a tangent direction $\hat{\mathbf{v}}$, which requires only the information provided by $\mathbf{F}(\mathbf{x}) = \mathbf{0}$ at $\mathbf{x}^0$.

In the following, we assume that the surface point $\mathbf{x}^0$ and its projection $\hat{\mathbf{x}}^0$ are nonsingular. Let $T_{\mathbf{x}^0}(S_F)$ denote the affine tangent space to S_F at $\mathbf{x}^0$, that is, the set of all tangent vectors to surface S_F at $\mathbf{x}^0$. It is evident that $T_{\mathbf{x}^0}(S_F)$ is the null space of $D\mathbf{F}(\mathbf{x}^0)\ \mathbf{h} = \mathbf{0}$, and that $T_{\mathbf{x}^0}(S_F)$ is a vector space of dimension 2 since $D\mathbf{F}(\mathbf{x}^0)$ has rank $n-2$. Note that at nonsingular surface points the tangent space

of the surface S_f at $\hat{\mathbf{x}}^0$, denoted as $T_{\hat{\mathbf{x}}^0}(S_f)$, has the same dimension as $T_{\mathbf{x}^0}(S_F)$. That is, the dimensionality of the tangent space is then invariant under projection.

A vector $\mathbf{n}$ is a normal to surface S_F at $\mathbf{x}^0$ if $\mathbf{n} \cdot \mathbf{h} = \mathbf{0}$ for every $\mathbf{h} \in T_{\mathbf{x}^0}(S_F)$. The normal vectors form a vector space of dimension $n-2$. Since $D\mathbf{F}(\mathbf{x}^0)$ has maximal rank $n-2$, the gradient vectors $\nabla f_1(\mathbf{x}^0), \ldots, \nabla f_{n-2}(\mathbf{x}^0)$ form a basis for the normal vector space of surface S_F at $\mathbf{x}^0$. Thus any linear combination of the gradients vectors is a vector in the normal space. Note that the normal space of surface S_f at a nonsingular surface point is of dimension 1.

3.1.1 Normal Vector

One may expect that computing the normal vector of the projected surface S_f at point $\hat{\mathbf{x}}^0 = (x_1^0, x_2^0, x_3^0)$ from the $D\mathbf{F}(\mathbf{x}^0)$ is as complex as the elimination process from $\mathbf{F}(\mathbf{x}) = \mathbf{0}$ to $f(x_1, x_2, x_3) = 0$. Indeed, it is true unless we approach the computation differently. Instead of considering the global surface S_F, we consider its tangent space at $\mathbf{x}^0$ and the tangent plane of S_f at $\hat{\mathbf{x}}^0$. The tangent space of S_F at $\mathbf{x}^0$ is the null space of

$$D\mathbf{F}(\mathbf{x}^0)\,(\mathbf{x} - \mathbf{x}^0) = \mathbf{0} \tag{3.3}$$

from which the equation of the tangent plane of S_f at $\hat{\mathbf{x}}^0$

$$Ax_1 + Bx_2 + Cx_3 + D = 0 \tag{3.4}$$

can be obtained by linear algebra techniques such as Gaussian elimination [43]. Considering the elimination process, we obtain the following algorithm for computing the normal vector of S_f at point $\hat{\mathbf{x}}^0$:

Algorithm 3.1

1. Consider $\mathbf{C} = \sum_{i=1}^{n-2} \alpha_i \nabla f_i(\mathbf{x}^0)$, where component C_i of $\mathbf{C}$ is a the linear combination of the unknowns α_i, $i = 1, \ldots, n-2$.

2. Solve the linear system

$$C_4 = 0, \quad C_5 = 0, \quad \ldots, \quad C_n = 0$$

which is of dimension $(n-3) \times (n-2)$ and has one degree of freedom.

3. Let $(\hat{\alpha}_1, \hat{\alpha}_2, \ldots, \hat{\alpha}_{n-2})$ be a solution of the above system and $\hat{\mathbf{C}}$ be the corresponding $\mathbf{C}$ in step 1. Then the first three components of $\hat{\mathbf{C}}$ are the normal vector of S_f at the regular point $\hat{\mathbf{x}}^0$.

Theorem 3.1 *Algorithm 3.1 computes the normal vector of the projected surface S_f at $\hat{\mathbf{x}}^0 = (x_1^0, x_2^0, x_3^0)$ if $\hat{\mathbf{x}}^0$ is a nonsingular point on S_f.*

Proof: Let I be the *ideal* generated by the system (3.3). Then

$$C(\mathbf{x}) = \sum_{i=1}^{n-2} \alpha_i \nabla f_i(\mathbf{x}^0)(\mathbf{x} - \mathbf{x}^0)$$

lies in the ideal I. Since (3.3) is a linear system, and the tangent plane (3.4) of S_f at $\hat{\mathbf{x}}^0$ can be obtained by eliminating the variables $x_4, x_5, \ldots, x_n$, step 2 of the algorithm is the elimination of the variables $x_4, x_5, \ldots, x_n$. Therefore, the $\hat{C}(\mathbf{x})$ with the computed coefficients $(\hat{\alpha}_1, \hat{\alpha}_2, \ldots, \hat{\alpha}_{n-2})$ is in $I \cap K[x_1, x_2, x_3]$. Hence the vector consisting of the first three components of $\hat{\mathbf{C}}$ is the gradient of $\hat{C}(\mathbf{x})$ which is the normal vector of S_f at $\hat{\mathbf{x}}^0$ corresponding to the normal space of S_F at $\mathbf{x}^0$. ■

Note that this computation is valid only when $\hat{\mathbf{x}}^0$ is nonsingular. Suppose the computation is valid at the singular point $\hat{\mathbf{x}}^0$. Then the matrix after applying Gaussian elimination to $D\mathbf{F}(\mathbf{x}^0)$ will not have maximal rank, which contradicts to the nonsingularity assumption of $\mathbf{x}^0$.

3.1.2 Tangent Vectors

Since the affine tangent space $T_{\mathbf{x}^0}(S_F)$ of S_F at a nonsingular surface point $\mathbf{x}^0$ has dimension two and is the null space of $D\mathbf{F}(\mathbf{x}^0)\,\mathbf{h} = 0$, we let $\mathbf{h}_1$ and $\mathbf{h}_2$ be a basis of $T_{\mathbf{x}^0}(S_F)$. The following theorem shows that the corresponding basis of

$T_{\hat{\mathbf{x}}^0}(S_f)$ are the natural projections of $\mathbf{h}_1$ and $\mathbf{h}_2$, denoted as $\hat{\mathbf{h}}_1$ and $\hat{\mathbf{h}}_2$ respectively, provided that $\hat{\mathbf{h}}_1$ and $\hat{\mathbf{h}}_2$ are linearly independent.

Theorem 3.2 *If* $\hat{\mathbf{h}}_1$ *and* $\hat{\mathbf{h}}_2$ *are the natural projections of* $\mathbf{h}_1$ *and* $\mathbf{h}_2$*, respectively, into* (x_1, x_2, x_3)*-space and are linearly independent, then* $\hat{\mathbf{h}}_1$ *and* $\hat{\mathbf{h}}_2$ *form a basis of* $T_{\hat{\mathbf{x}}^0}(S_f)$.

Proof: Since $D\mathbf{F}(\mathbf{x}^0)$ has maximal rank $n-2$, its row-echelon normal form is

$$\begin{bmatrix} A & B & C & 0 & 0 & 0 & \cdots & 0 \\ \times & \times & \times & * & 0 & 0 & \cdots & 0 \\ \times & \times & \times & \times & * & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \times & \times & \times & \times & \times & \times & \cdots & * \end{bmatrix}$$

where A, B, and C are coefficients of linear terms in (3.4), $*$'s are nonzero numbers and the $\times$'s represent numbers that may or may not be zero. Since $\mathbf{h}_1$ and $\mathbf{h}_2$ form a basis of $T_{\mathbf{x}^0}(S_F)$ $\mathbf{h}_1$ and $\mathbf{h}_2$ are linearly independent and satisfy

$$[A, B, C, 0, 0, 0, \cdots, 0]\,\mathbf{h} = 0$$

Thus $\hat{\mathbf{h}}_1$ and $\hat{\mathbf{h}}_2$ satisfy

$$[A, B, C]\,\hat{\mathbf{h}} = 0$$

Furthermore, $\hat{\mathbf{h}}_1$ and $\hat{\mathbf{h}}_2$ are linearly independent. Hence $\hat{\mathbf{h}}_1$ and $\hat{\mathbf{h}}_2$ form a basis of $T_{\hat{\mathbf{x}}^0}(S_f)$. ■

For a given tangent $\mathbf{h}$ to S_F at a nonsingular surface point $\mathbf{x}^0$ the tangent to S_f at $\hat{\mathbf{x}}^0$ is the natural projection of $\mathbf{h}$ assuming $\hat{\mathbf{x}}^0$ is nonsingular.

Example 3.1 Consider the general parametric surface

$$\begin{aligned} x &= h_1(s,t) \\ y &= h_2(s,t) \\ z &= h_3(s,t) \end{aligned}$$

and a surface point $(x^0, y^0, z^0) = (h_1(s^0, t^0), h_2(s^0, t^0), h_3(s^0, t^0))$. In five dimensional space, we write the surface as

$$\begin{aligned} f_1(x, y, z, s, t) &= x - h_1(s, t) &= 0 \\ f_2(x, y, z, s, t) &= y - h_2(s, t) &= 0 \\ f_3(x, y, z, s, t) &= z - h_3(s, t) &= 0 \end{aligned}$$

with surface point $\mathbf{x}^0 = (x^0, y^0, z^0, s^0, t^0)$. The gradients of the f_i are

$$\begin{aligned} \nabla f_1(x^0, y^0, z^0, s^0, t^0) &= (1, 0, 0, h_{1,s}, h_{1,t}) \\ \nabla f_2(x^0, y^0, z^0, s^0, t^0) &= (0, 1, 0, h_{2,s}, h_{2,t}) \\ \nabla f_3(x^0, y^0, z^0, s^0, t^0) &= (0, 0, 1, h_{3,s}, h_{3,t}) \end{aligned}$$

where $h_{i,s}$ and $h_{i,t}$ are partial derivatives of h_i with respect to s and t respectively and evaluated at (s^0, t^0).

The linear system

$$\begin{aligned} C_4 &= \alpha_1 h_{1,s} + \alpha_2 h_{2,s} + \alpha_3 h_{3,s} &= 0 \\ C_5 &= \alpha_1 h_{1,t} + \alpha_2 h_{2,t} + \alpha_3 h_{3,t} &= 0 \end{aligned}$$

has a solution

$$(\hat{\alpha}_1, \hat{\alpha}_2, \hat{\alpha}_3) = (h_{2s}h_{3t} - h_{2t}h_{3s}, h_{1t}h_{3s} - h_{1s}h_{3t}, h_{1s}h_{2t} - h_{1t}h_{2s})$$

which is $(h_{1s}, h_{2s}, h_{3s}) \times (h_{1t}, h_{2t}, h_{3t})$. Thus, the corresponding $\hat{\mathbf{C}}$ as defined in the algorithm is $(\hat{\alpha}_1, \hat{\alpha}_2, \hat{\alpha}_3, 0, 0)$ and $(\hat{\alpha}_1, \hat{\alpha}_2, \hat{\alpha}_3)$ is the normal vector of the surface at (x^0, y^0, z^0).

The $D\mathbf{F}(\mathbf{x}^0)\,\mathbf{h} = \mathbf{0}$ has the general solution

$$\lambda_1(h_{1s}, h_{2s}, h_{3s}, -1, 0) + \lambda_2(h_{1t}, h_{2t}, h_{3t}, 0, -1)$$

Thus the basis vectors of the affine tangent space are

$$\mathbf{h}_1 = (h_{1s}, h_{2s}, h_{3s}, -1, 0) \text{ and } \mathbf{h}_2 = (h_{1t}, h_{2t}, h_{3t}, 0, -1)$$

and their projections into (x_1, x_2, x_3)-space

$$\hat{\mathbf{h}}_1 = (h_{1s}, h_{2s}, h_{3s}) \text{ and } \hat{\mathbf{h}}_2 = (h_{1t}, h_{2t}, h_{3t})$$

form the basis of the corresponding tangent space in (x, y, z)-space. ■

Example 3.2 Consider the implicit surface $g(x, y, z) = 0$ and its offset given in Example 1.1. $D\mathbf{F}(\mathbf{x}^0)$ of (1.4) at surface point $\mathbf{x}^0 = (x^0, y^0, z^0, u^0, v^0, w^0) = (0, 0, 4, 0, 0, 2)$ is

$$\begin{bmatrix} 0 & 0 & 4 & 0 & 0 & -4 \\ 0 & 0 & 0 & 0 & 0 & 4 \\ 2 & 0 & 0 & -10 & 0 & 0 \\ 0 & 2 & 0 & 0 & -4 & 0 \end{bmatrix}$$

The linear system with $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ as unknowns is

$$\begin{aligned} C_4 &= -10\alpha_3 = 0 \\ C_5 &= -4\alpha_4 = 0 \\ C_6 &= 4\alpha_2 - 4\alpha_1 = 0 \end{aligned}$$

and has a solution $(\hat{\alpha}_1, \hat{\alpha}_2, \hat{\alpha}_3, \hat{\alpha}_4) = (1, 1, 0, 0)$. Hence the corresponding $\hat{\mathbf{C}}$ in the algorithm is $(0, 0, 4, 0, 0, 0)$ in which $(0, 0, 4)$ is the normal vector at (x^0, y^0, z^0).

The general solution of $D\mathbf{F}(\mathbf{x}^0)\ \mathbf{h} = \mathbf{0}$ is $\lambda_1\mathbf{h}_1 + \lambda_2\mathbf{h}_2$ where

$$\mathbf{h}_1 = (1, 0, 0, 1/5, 0, 0)$$

and

$$\mathbf{h}_2 = (0, 2, 0, 0, 1, 0)$$

serve as the basis of the affine tangent space. By natural projection,

$$\hat{\mathbf{h}}_1 = (1, 0, 0)$$

and

$$\hat{\mathbf{h}}_2 = (0, 2, 0)$$

are the basis vectors of the corresponding tangent space in (x, y, z)-space. ■

3.1.3 Normal Curvature

For a 2-surface in $\mathbf{R}^3$, formulae for computing the normal curvature of the surface along a tangent direction at a surface point can be found in most differential geometry books. However, we wish to compute the normal curvature of the projected surface S_f, in (x_1, x_2, x_3)-space, from a given 2-surface $\mathbf{F}(\mathbf{x}) = \mathbf{0}$ in $\mathbf{R}^n$. It is clear that one could first determine the local explicitization or local parameterization (both are in (x_1, x_2, x_3)-space), discussed later in subsequent sections, and then apply the standard formulae. In this section, we describe a method that directly computes the normal curvature of the surface S_f from the high-dimensional surface representation $\mathbf{F}(\mathbf{x}) = \mathbf{0}$, without constructing a local approximant first.

3.1.3.1 The Normal Curvature of Hypersurface in $\mathbf{R}^n$

Let $g(x_1, x_2, \ldots, x_n) = 0$ be a hypersurface in $\mathbf{R}^n$ and let S_g represent the zero set of $g = 0$. For $p \in S_g$ and $\mathbf{v} \in T_p(S_g)$, and a normal vector field N on S_g, we define the linear map $\mathbf{L}_p : T_p(S_g) \rightarrow T_p(S_g)$ as

$$\mathbf{L}_p(\mathbf{v}) = -\nabla_{\mathbf{v}} N = -(N \circ \beta)'(0)$$

where $\beta : I \rightarrow S_g$ is any parametrized curve on S_g with $\beta(0) = p$ and $\dot{\beta}(0) = \mathbf{v}$, and $(N \circ \beta)(t) = N(\beta(t))$. The definition makes sense since the directional derivative $\nabla_{\mathbf{v}} N$ is tangent to S_g. $\mathbf{L}_p$ is usually called the *Weingarten map* or *shape operator* of S_g at p, see, e.g., [50, 66]. The generalization of Meusnier's theorem to high dimensions states that $\ddot{\beta}(0) \cdot N(p)$ is invariant for all parameterized curves β on S_g with $\beta(0) = p$ and $\dot{\beta}(0) = \mathbf{v}$.

Lemma 3.1 (Meusnier) *Let β be a parameterized curve on the hypersurface g in $\mathbf{R}^n$ with $\beta(0) = p$ and $\dot{\beta}(0) = \mathbf{v}$. Then*

$$\mathbf{L}_p(\mathbf{v}) \cdot \mathbf{v} = \ddot{\beta}(0) \cdot N(p)$$

where N is a normal vector field on S_g.

If N is the unit normal vector field on S_g and $\mathbf{v}$ is a unit tangent vector of S_g at p, the *normal curvature* $k_g(\mathbf{v})$ of S_g at p in the direction $\mathbf{v}$ is defined as

$$k_g(\mathbf{v}) = \mathbf{L}_p(\mathbf{v}) \cdot \mathbf{v}$$

The computation of $\mathbf{L}_p(\mathbf{v}) \cdot \mathbf{v}$ can be conducted according to the following lemma:

Lemma 3.2 *Let $N = \nabla g$, H_g be the Hessian matrix of g at p, and $\mathbf{v}$ be a tangent vector of g at p. Then*

$$\mathbf{L}_p(\mathbf{v}) \cdot \mathbf{v} = -\mathbf{v}^T H_g \mathbf{v}$$

Moreover, when $\mathbf{v}$ is a unit tangent

$$k_g(\mathbf{v}) = \frac{1}{\|\nabla g(p)\|} \mathbf{L}_p(\mathbf{v}) \cdot \mathbf{v}$$

Proof:

$$\begin{aligned}
\mathbf{L}_p(\mathbf{v}) \cdot \mathbf{v} &= -\nabla_{\mathbf{v}} N \cdot \mathbf{v} \\
&= -\nabla_{\mathbf{v}} \nabla g \cdot \mathbf{v} \\
&= -\left(\nabla_{\mathbf{v}} \frac{\partial g}{\partial x_1}, \nabla_{\mathbf{v}} \frac{\partial g}{\partial x_2}, \ldots, \nabla_{\mathbf{v}} \frac{\partial g}{\partial x_n} \right) \cdot \mathbf{v} \\
&= -\left[\nabla \left(\frac{\partial g}{\partial x_1} \right)(p) \cdot \mathbf{v}, \nabla \left(\frac{\partial g}{\partial x_2} \right)(p) \cdot \mathbf{v}, \ldots, \nabla \left(\frac{\partial g}{\partial x_n} \right)(p) \cdot \mathbf{v} \right] \cdot \mathbf{v} \\
&= -\left[\sum_{i=1}^{n} \frac{\partial^2 g}{\partial x_i \partial x_1}(p) v_i, \sum_{i=1}^{n} \frac{\partial^2 g}{\partial x_i \partial x_2}(p) v_i, \ldots, \sum_{i=1}^{n} \frac{\partial^2 g}{\partial x_i \partial x_n}(p) v_i \right] \cdot \mathbf{v} \\
&= -\sum_{i,j=1}^{n} \frac{\partial^2 g}{\partial x_i \partial x_j}(p) v_i v_j \\
&= -\mathbf{v}^T H_g \mathbf{v}
\end{aligned}$$

According to Meusnier's Theorem, when $\mathbf{v}$ is a unit tangent we have

$$k_g(\mathbf{v}) = \ddot{\beta}(0) \cdot \frac{N(p)}{\|\nabla g(p)\|}$$

where β is a parameterized curve on S_g with $\beta(0) = p$ and $\dot{\beta}(0) = \mathbf{v}$. Thus

$$k_g(\mathbf{v}) = \frac{1}{\|\nabla g(p)\|} \mathbf{L}_p(\mathbf{v}) \cdot \mathbf{v}$$

■

3.1.3.2 The Normal Curvature of Projected 2-D Surface in $\mathbf{R}^3$

For a 2-surface S_F in $\mathbf{R}^n$ and its projected 2-surface S_f in (x_1, x_2, x_3)-space, we intend to compute the normal curvature of S_f in a given tangent direction directly from $\mathbf{F}(\mathbf{x}) = \mathbf{0}$ without computing f.

Let $\hat{\mathbf{x}}^0 \in S_f$ and $\hat{\mathbf{v}} \in T_{\hat{\mathbf{x}}^0}(S_f)$ be the projections of $\mathbf{x}^0 \in S_F$ and $\mathbf{v} \in T_{\mathbf{x}^0}(S_F)$, respectively, into (x_1, x_2, x_3)-space. In addition, let N_i be the normal vector field on $f_i = 0$. Since $\mathbf{x}^0 \in S_F$ and $\mathbf{v}$ is a tangent to S_F at $\mathbf{x}^0$, $\mathbf{x}^0$ is a surface point of $f_i = 0$ and $\mathbf{v}$ is a tangent vector to $f_i = 0$ at $\mathbf{x}^0$ for $i = 1, 2, \ldots, n-2$.

Let $\alpha_1, \ldots, \alpha_{n-2}$ be real numbers such that $\sum_{i=1}^{n-2} \alpha_i N_i(\mathbf{x}^0) = (A, B, C, 0, \ldots, 0)$, where $Ax + By + Cz + D = 0$ is the tangent plane of $f = 0$ at $\hat{\mathbf{x}}^0$, i.e., (A, B, C) is the normal vector of S_f at $\hat{\mathbf{x}}^0$. In the following, we will show that the $\mathbf{L}^i_{\mathbf{x}^0}(\mathbf{v}) \cdot \mathbf{v}$, where $\mathbf{L}^i_{\mathbf{x}^0}(\mathbf{v}) = -\nabla_{\mathbf{v}} N_i$, for $i = 1, \ldots, n-2$, and the linear combination $\sum_{i=1}^{n-2} \alpha_i (\mathbf{L}^i_{\mathbf{x}^0}(\mathbf{v}) \cdot \mathbf{v})$ play an important role in computing the normal curvature of $f(x_1, x_2, x_3) = 0$.

Consider any curve $\beta(t) = (\beta_1(t), \ldots, \beta_n(t))$ on S_F with $\beta(0) = \mathbf{x}^0$ and $\dot{\beta}(0) = \mathbf{v}$. Then β is a surface curve on $f_i = 0$ as well, for $i = 1, 2, \ldots, n-2$. By Meusnier's theorem we have

$$\mathbf{L}^i_{\mathbf{x}^0}(\mathbf{v}) \cdot \mathbf{v} = <\ddot{\beta}(0), N_i(\mathbf{x}^0)> \tag{3.5}$$

Hence

$$\begin{aligned}
\sum_{i=1}^{n-2} \alpha_i (\mathbf{L}^i_{\mathbf{x}^0}(\mathbf{v}) \cdot \mathbf{v}) &= \sum_{i=1}^{n-2} \alpha_i <\ddot{\beta}(0), N_i(\mathbf{x}^0)> \\
&= \sum_{i=1}^{n-2} <\ddot{\beta}(0), \alpha_i N_i(\mathbf{x}^0)> \\
&= <\ddot{\beta}(0), \sum_{i=1}^{n-2} \alpha_i N_i(\mathbf{x}^0)> \\
&= <\ddot{\beta}(0), (A, B, C, 0, \ldots, 0)> \\
&= <(\ddot{\beta}_1(0), \ddot{\beta}_2(0), \ddot{\beta}_3(0)), (A, B, C)> \qquad (3.6)
\end{aligned}$$

where $(\beta_1(t), \beta_2(t), \beta_3(t))$ is the projected curve of β on S_f with

$$(\beta_1(0), \beta_2(0), \beta_3(0)) = \hat{\mathbf{x}}^0 \text{ and } (\dot{\beta}_1(0), \dot{\beta}_2(0), \dot{\beta}_3(0)) = \hat{\mathbf{v}}$$

and (A, B, C) is a normal vector of S_f at $\hat{\mathbf{x}}^0$. Notice that (A, B, C) and $\hat{\mathbf{v}}$ may not be a unit normal and a unit tangent, respectively, to S_f at $\hat{\mathbf{x}}^0$. For the right hand side of (3.6) to represent the normal curvature of S_f, both the normal vector (A, B, C) of S_f at $\hat{\mathbf{x}}^0$ and $(\dot{\beta}_1(0), \dot{\beta}_2(0), \dot{\beta}_3(0))$ must be unit vectors. Thus, when β is a curve such that $(\dot{\beta}_1(0), \dot{\beta}_2(0), \dot{\beta}_3(0))$ is a unit vector, according to Meusnier's theorem, the right hand side of (3.6) divided by the length of (A, B, C) is the normal curvature of S_f at $\hat{\mathbf{x}}^0$ in the direction $(\dot{\beta}_1(0), \dot{\beta}_2(0), \dot{\beta}_3(0))$. Hence, the left hand side of (3.6) divided by $\|(A, B, C)\|$ is the normal curvature of S_f at $\hat{\mathbf{x}}^0$ in the direction of $\hat{\mathbf{v}}$, provided that $\hat{\mathbf{v}}$ is a unit tangent. Notice that the equality in (3.5) is valid for a non-unit normal vector field N_i and any tangent vector $\mathbf{v}$. We summarize this fact as follows:

Theorem 3.3 *Let $N_i = \nabla f_i$, $i = 1, 2, \ldots, n-2$ and let $\alpha_1, \ldots, \alpha_{n-2}$ be numbers such that $\sum_{i=1}^{n-2} \alpha_i N_i(\mathbf{x}^0) = (A, B, C, 0, \ldots, 0)$, where (A, B, C) is the normal of S_f at $\hat{\mathbf{x}}^0$. Also let $\mathbf{v}$ be a tangent vector of S_F at $\mathbf{x}^0$ such that $\hat{\mathbf{v}}$ is a unit tangent to S_f at $\hat{\mathbf{x}}^0$. Then*

$$\frac{1}{\|(A, B, C)\|} \sum_{i=1}^{n-2} \alpha_i (\mathbf{L}^i_{\mathbf{x}^0}(\mathbf{v}) \cdot \mathbf{v})$$

is the normal curvature of S_f at $\hat{\mathbf{x}}^0$ in the $\hat{\mathbf{v}}$ direction.

The theorem and its proof suggest to us two computation schemes:

Algorithm 3.2

1. Determine $\mathbf{v}$ such that $\hat{\mathbf{v}}$ is unit.
2. Derive $\beta(t) = (\beta_1(t), \ldots, \beta_n(t))$ on S_F such that $\beta(0) = \mathbf{x}^0$ and $\dot{\beta}(0) = \mathbf{v}$.
3. Compute $(\ddot{\beta}_1(0), \ddot{\beta}_2(0), \ddot{\beta}_3(0))$.
4. Compute $(A, B, C, 0, \ldots, 0) = \sum_{i=1}^{n-2} \alpha_i N_i(\mathbf{x}^0)$ for some $\alpha_1, \ldots, \alpha_{n-2}$ and normalize (A, B, C) to $(\hat{A}, \hat{B}, \hat{C})$ with unit length.
5. Compute $<(\ddot{\beta}_1(0), \ddot{\beta}_2(0), \ddot{\beta}_3(0)), (\hat{A}, \hat{B}, \hat{C})>$.

Algorithm 3.2 is conceptually very simple, however, the computation of the β on a 2-surface S_F is nontrivial.

Note that, as shown in Lemma 3.2, $\mathbf{L}^i_{\mathbf{x}^0}(\mathbf{v}) \cdot \mathbf{v} = -\mathbf{v}^T H_i\, \mathbf{v}$, where H_i is the Hessian of the hypersurface f_i at $\mathbf{x}^0$. Moreover, because of the bilinearity of the form $\mathbf{v}^T H_i\, \mathbf{v}$, we have

$$\sum_{i=1}^{n-2} \alpha_i (\mathbf{L}^i_{\mathbf{x}^0}(\mathbf{v}) \cdot \mathbf{v}) = \sum_{i=1}^{n-2} \alpha_i (-\mathbf{v}^T H_i\, \mathbf{v}) = -\mathbf{v}^T (\sum_{i=1}^{n-2} \alpha_i H_i)\mathbf{v}$$

In the following, we state an algorithm that is well suited to computing the normal curvature of S_f at $\hat{\mathbf{x}}^0$ in different tangent directions.

Algorithm 3.3

1. Compute $N_i = \nabla f_i$, for $i = 1, 2, \ldots, n-2$.
2. Compute $\alpha_1, \ldots, \alpha_{n-2}$ such that $\sum_{i=1}^{n-2} \alpha_i N_i(\mathbf{x}^0) = (A, B, C, 0, \ldots, 0)$, where (A, B, C) is the normal of S_f at $\hat{\mathbf{x}}^0$.
3. Compute $H_0 = \sum_{i=1}^{n-2} \alpha_i H_i$, where H_i is the Hessian of f_i at $\mathbf{x}^0$.
4. Adjust $\mathbf{v}$ such that $\hat{\mathbf{v}}$ is a unit tangent to S_f at $\hat{\mathbf{x}}^0$.
5. The normal curvature of S_f at $\hat{\mathbf{x}}^0$ in the $\hat{\mathbf{v}}$ direction is

$$\frac{-1}{\|(A, B, C)\|} \mathbf{v}^T H_0\, \mathbf{v}$$

Let $\mathbf{L}^f_{\hat{\mathbf{x}}^0}$ be the shape operator of the closed form $f(x_1, x_2, x_3) = 0$ at $\hat{\mathbf{x}}^0$. Due to Meusnier's theorem, from (3.6) we obtain a formula for computing $\mathbf{L}^f_{\hat{\mathbf{x}}^0}(\hat{\mathbf{v}}) \cdot \hat{\mathbf{v}}$.

Theorem 3.4 *Let $N_i = \nabla f_i$, $i = 1, 2, \ldots, n-2$ and let $\alpha_1, \ldots, \alpha_{n-2}$ be numbers such that $\sum_{i=1}^{n-2} \alpha_i N_i(\mathbf{x}^0) = (A, B, C, 0, \ldots, 0)$, where (A, B, C) is the normal of S_f at $\hat{\mathbf{x}}^0$. Also let $\mathbf{v}$ be a tangent vector of S_F at $\mathbf{x}^0$. Then*

$$\mathbf{L}^f_{\hat{\mathbf{x}}^0}(\hat{\mathbf{v}}) \cdot \hat{\mathbf{v}} = \sum_{i=1}^{n-2} \alpha_i (\mathbf{L}^i_{\mathbf{x}^0}(\mathbf{v}) \cdot \mathbf{v})$$

Example 3.3 Consider the implicit surface $g(x,y,z) = 0$ and its offset given in Example 1.1. Let N_i be the gradient for the i-th polynomial f_i, $i = 1,2,3,4$, $\mathbf{x}^0 = (x^0, y^0, z^0, u^0, v^0, w^0) = (0,0,4,0,0,2)$, and $\mathbf{v} = (0,1,0,0,1/2,0)$ be a tangent vector at $\mathbf{x}^0$. In Example 3.2, we have computed $(\alpha_1, \alpha_2, \alpha_3, \alpha_4) = (1,1,0,0)$ such that

$$(0,0,4,0,0,0) = \sum_{i=1}^{4} \alpha_i N_i(\mathbf{x}^0)$$

Now, we compute $H_0 = \sum_{i=1}^{4} \alpha_i H_i$ and obtain

$$H_0 = \begin{bmatrix} 2 & 0 & 0 & -2 & 0 & 0 \\ 0 & 2 & 0 & 0 & -2 & 0 \\ 0 & 0 & 2 & 0 & 0 & -2 \\ -2 & 0 & 0 & 10 & 0 & 0 \\ 0 & -2 & 0 & 0 & 4 & 0 \\ 0 & 0 & -2 & 0 & 0 & 4 \end{bmatrix}$$

Thus,

$$\frac{-1}{\|(0,0,4)\|}\mathbf{v}^T H_0\, \mathbf{v} = -1/4$$

which is expected to be the normal curvature of S_f at $(0,0,4)$ in the direction of $(0,1,0)$. Intersecting the projected surface with the plane that goes through $(0,0,4)$ and is spanned by the normal $(0,0,4)$ and tangent $(0,1,0)$, we find that the normal section is a circle of radius 4 and has normal curvature $-1/4$ at $(0,0,4)$ in the direction of $(0,1,0)$, which verifies the result. ■

Example 3.4 Consider the Voronoi surface of g and h in Example 1.2. Let N_i be the gradient for i-th polynomial f_i, $i = 1,2,\ldots,8$, and let

$$\mathbf{x}^0 = (x^0, y^0, z^0, u^0, v^0, w^0, \bar{u}^0, \bar{v}^0, \bar{w}^0, r^0) = (0,0,0,0,0,1,0,0,-1,1)$$

and $\mathbf{v} = (0,1,0,0,1,0,0,1/2,0,0)$ be a tangent vector at $\mathbf{x}^0$. We find

$$(\alpha_1, \alpha_2, \ldots, \alpha_8) = (1,1,0,0,-1,-1,0,0)$$

such that

$$(0,0,-4,0,\ldots,0) = \sum_{i=1}^{8} \alpha_i N_i(\mathbf{x}^0)$$

We then compute $H_0 = \sum_{i=1}^{8} \alpha_i H_i$ and obtain

$$H_0 = \begin{bmatrix} 0 & 0 & 0 & -2 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -2 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -2 & 0 & 0 & 2 & 0 \\ -2 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -2 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -2 & 0 & 0 & 4 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 & -4 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & -4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Thus,

$$\frac{-1}{\|(0,0,-4)\|}\mathbf{v}^T H_0 \mathbf{v} = 1/4$$

which is expected to be the normal curvature of S_f at $(0,0,0)$ in the direction of $(0,1,0)$. As shown in Example 1.2, the even Voronoi surface of g and h is $x^2 - y^2 - 8z$ which does have normal curvature $1/4$ at $(0,0,0)$ in the direction $(0,1,0)$. ■

3.2 Degree Two Implicit Approximation

Our task here is to derive a degree two implicit approximant of S_f in (x_1, x_2, x_3)-space for the given $\mathbf{F}(\mathbf{x}) = 0$ and a surface point $\mathbf{x}^0$. We require that the approximant has the same normal direction as S_f at $\mathbf{x}^0$ and that its normal curvatures agree with those of S_f in all tangent directions at $\mathbf{x}^0$. The constraint on curvature agreement in all directions is difficult to realize straightforwardly. However, using the following *Three Tangents Theorem* [52], we are able to formulate the constraint easily.

Theorem 3.5 (Three Tangents Theorem) *Two surfaces with common normal direction at a surface point have the same normal curvatures in every tangent direction if and only if their normal curvatures agree in three tangent directions of which any two are linearly independent.*

Applying this theorem, the constraints above can be translated into a system of seven equations, some linear and some nonlinear. The nonlinear equations stem from the normalization of the normal vector of the approximant in the normal curvature computation. Using Lemma 3.2 to rephrase the Three Tangents Theorem, however, results in a system of linear equations which enforce the constraints.

Corollary 3.1 *Two surfaces with identical normal vector at a surface point p have the same normal curvatures in every tangent direction if and only if values of their* $\mathbf{L}_p(\mathbf{v})\cdot\mathbf{v}$ *agree in three tangent directions of which any two are linearly independent.*

Let $g^2(x_1, x_2, x_3) = \sum_{i+j+k=0}^{2} a_{ijk} x_1^i x_2^j x_3^k$ be a generic degree two polynomial with the symbolic coefficients a_{ijk}. Suppose $\mathbf{v}_1, \mathbf{v}_2$ and $\mathbf{v}_3$ are tangents to S_F at $\mathbf{x}^0$ with the property that any pair of their projections, $\hat{\mathbf{v}}_1, \hat{\mathbf{v}}_2$ and $\hat{\mathbf{v}}_3$ respectively, in (x_1, x_2, x_3)-space is linearly independent. We translate the constraints to a linear system as follows:

1. $g^2(\hat{\mathbf{x}}^0) = 0$.

2. $\nabla g^2(\hat{\mathbf{x}}^0) = (A, B, C)$, where (A, B, C) is the normal vector of S_f at $\hat{\mathbf{x}}^0$ computed using Algorithm 3.1.

3. For $i = 1, 2, 3$,

$$\mathbf{L}^g_{\hat{\mathbf{x}}^0}(\hat{\mathbf{v}}_i)\cdot\hat{\mathbf{v}}_i = \mathbf{L}^f_{\hat{\mathbf{x}}^0}(\hat{\mathbf{v}}_i)\cdot\hat{\mathbf{v}}_i$$

   where $\mathbf{L}^g_{\hat{\mathbf{x}}^0}(\hat{\mathbf{v}}_i)\cdot\hat{\mathbf{v}}_i = -\hat{\mathbf{v}}_i^T H_g \hat{\mathbf{v}}_i$ and $\mathbf{L}^f_{\hat{\mathbf{x}}^0}(\hat{\mathbf{v}}_i)\cdot\hat{\mathbf{v}}_i$ is computed according to Theorem 3.4.

This formulation results in 7 linear equations in 10 variables. Since $g^2(x_1, x_2, x_3) = 0$ and $\gamma g^2(x_1, x_2, x_3) = 0$, where $\gamma \neq 0$, represent the same surface, $g^2(x_1, x_2, x_3)$

has 9 coefficients on which the surface depends. Thus, we can adjoin to the linear system the additional equation

$$a_{ijk} - 1 = 0$$

for some $0 \leq i + j + k \leq 2$, and obtain a 8×10 linear system from which the coefficients of g^2 can be computed with two degrees of freedom. We give the algorithm as follows:

Algorithm 3.4

1. Form a generic polynomial $g^2(x_1, x_2, x_3) = \sum_{i+j+k=0}^{2} a_{ijk} x_1^i x_2^j x_3^k$, where the a_{ijk} are symbolic coefficients.
2. Compute $N_i = \nabla f_i$, for $i = 1, 2, \ldots, n-2$.
3. Compute $\alpha_1, \ldots, \alpha_{n-2}$ such that $\sum_{i=1}^{n-2} \alpha_i N_i(\mathbf{x}^0) = (A, B, C, 0, \ldots, 0)$, where (A, B, C) is the normal of S_f at $\hat{\mathbf{x}}^0$.
4. Compute $H_0 = \sum_{i=1}^{n-2} \alpha_i H_i$, where H_i is the Hessian of f_i at $\mathbf{x}^0$.
5. Derive three tangent vectors $\mathbf{v}_1, \mathbf{v}_2$ and $\mathbf{v}_3$ to S_F at $\mathbf{x}^0$ such that any pair of their projections, $\hat{\mathbf{v}}_1, \hat{\mathbf{v}}_2$ and $\hat{\mathbf{v}}_3$ respectively, in (x_1, x_2, x_3)-space is linearly independent.
6. Let $\nabla g^2(\hat{\mathbf{x}}^0) = (g^2_{x_1}(\hat{\mathbf{x}}^0), g^2_{x_2}(\hat{\mathbf{x}}^0), g^2_{x_3}(\hat{\mathbf{x}}^0))$.
7. Form the following linear system:

$$\begin{aligned}
a_{ijk} - 1 &= 0, \qquad \text{for some } 0 \leq i + j + k \leq 2 \\
g^2(\hat{\mathbf{x}}^0) &= 0 \\
g^2_{x_1}(\hat{\mathbf{x}}^0) - A &= 0 \\
g^2_{x_2}(\hat{\mathbf{x}}^0) - B &= 0 \\
g^2_{x_3}(\hat{\mathbf{x}}^0) - C &= 0 \\
\hat{\mathbf{v}}_1^T H_g \hat{\mathbf{v}}_1 - \mathbf{v}_1^T H_0 \mathbf{v}_1 &= 0
\end{aligned}$$

$$\begin{aligned}\hat{\mathbf{v}}_2^T H_g \hat{\mathbf{v}}_2 - \mathbf{v}_2^T H_0 \mathbf{v}_2 &= 0\\ \hat{\mathbf{v}}_3^T H_g \hat{\mathbf{v}}_3 - \mathbf{v}_3^T H_0 \mathbf{v}_3 &= 0\end{aligned}$$

8. Solve the above system for the coefficients of g^2.

Example 3.5 Consider the implicit surface $g(x,y,z) = 0$ and its offset given in Example 1.1. Let $\mathbf{x}^0 = (x^0, y^0, z^0, u^0, v^0, w^0) = (0,0,4,0,0,2)$, and let $\mathbf{v}_1 = (1,2,0,1/5,1,0)$, $\mathbf{v}_2 = (1,1,0,1/5,1/2,0)$, $\mathbf{v}_3 = (2,3,0,2/5,3/2,0)$ be three tangents at $\mathbf{x}^0$. Note that any pair of their projections is linearly independent. Let $g^2(x_1,x_2,x_3) = \sum_{i+j+k=0}^{2} a_{ijk} x_1^i x_2^j x_3^k$ be a generic degree two polynomial with the symbolic coefficients a_{ijk}. We form the following linear system:

$$\begin{aligned}16a_{002} + 4a_{001} + a_{000} &= 0\\ 4a_{101} + a_{100} &= 0\\ 4a_{011} + a_{010} &= 0\\ 8a_{002} + a_{001} - 4 &= 0\\ -2a_{200} - 2a_{110} - 8a_{020} - 2a_{110} + 28/5 &= 0\\ -2a_{200} - 2a_{110} - 2a_{020} + 13/5 &= 0\\ -8a_{200} - 6a_{110} - 6a_{110} - 18a_{020} + 77/5 &= 0\end{aligned}$$

When $a_{000} - 1 = 0$ is added to the system, for the unknown coefficients

$$(a_{000}, a_{100}, a_{010}, a_{001}, a_{200}, a_{110}, a_{101}, a_{020}, a_{011}, a_{002})$$

we obtain the general solution

$$(1, -4\alpha, -4\beta, -9/2, 4/5, 0, \alpha, 1/2, \beta, 17/16)$$

With $\alpha = \beta = 1$, we have

$$g^2(x,y,z) = 1 - 4x - 4y - \frac{9}{2}z + \frac{4}{5}x^2 + xz + \frac{1}{2}y^2 + yz + \frac{17}{16}z^2$$

■

3.3 Local Explicit Approximation

The implicit function theorem ensures that system (3.1) determines $n-2$ components of $\mathbf{x}$ as functions of the remaining 2 components in a neighborhood of any surface point $\mathbf{x}^0$ at which $D\mathbf{F}(\mathbf{x}^0)$ has rank $n-2$. Since $D\mathbf{F}(\mathbf{x}^0)$ has maximal rank $n-2$, some set of $n-2$ columns of its matrix is linearly independent. In the following, without loss of generality, we assume that the last $n-2$ columns are linearly independent, and hence the determinant of the Jacobian matrix of $f_1, f_2, \ldots, f_{n-2}$ with respect to $x_3, x_4, \ldots, x_n$, denoted as $|J\mathbf{F}(\mathbf{x}^0)|$, is nonzero. The implicit function theorem guarantees that there exists a neighborhood V of $\mathbf{x}^0$, an open set $U \subset E^2$ containing (x_1^0, x_2^0), and a mapping $\Psi = (\psi_3, \psi_4, \ldots, \psi_n)$ defined on U such that $|J\mathbf{F}(\mathbf{x})| \neq 0$ for all $\mathbf{x} \in V$, and for $1 \leq i \leq n-2$,

$$f_i(x_1, x_2, \psi_3(x_1, x_2), \ldots, \psi_n(x_1, x_2)) = 0 \tag{3.7}$$

for all $(x_1, x_2) \in U$. Let ψ_i be defined as

$$\psi_i(x_1, x_2) = \sum_{j+k \geq 0} b_{jk}^{(i)} x_1^j x_2^k \tag{3.8}$$

and $\tilde{f}_i(x_1, x_2)$ denote $f_i(x_1, x_2, \psi_3(x_1, x_2), \ldots, \psi_n(x_1, x_2))$.

It is clear that, in (x_1, x_2, x_3)-space, $x_3 = \psi_3(x_1, x_2)$ represents the local explicit approximation of the projected surface (3.2) at (x_1^0, x_2^0, x_3^0), and might be called a *local explicitization* of the surface. The local explicitization of an implicit surface $g(x_1, x_2, x_3) = 0$ has been considered in [48].

The unknown coefficients of ψ_i can be calculated from the chain rule with a linear system solver. By requiring that the k-th derivatives of $\tilde{f}_i$, $i = 1, 2, \ldots, n-2$, with respect to x_1 and x_2 are identically zero, a linear system with unknown coefficients of degree k terms is obtained. It is structurally very simple. However, the direct application of the chain rule results in a formula which is algebraically tedious. In the following, we develop a recursive formulation which presents the computation in a more suitable manner.

When we assume that $\mathbf{x}^0 = (0, 0, \ldots, 0)$ is the origin, the constant terms of f_i, $a^{(i)}_{0,0,\ldots,0}$, and the constant term of ψ_i, $b^{(i)}_{00}$, are both zero and the partials $\frac{\partial \tilde{f}_i^{j+k}}{\partial x_1^j \partial x_2^k}$ evaluated at $(0,0)$ and divided by $i!$ and $j!$ are the coefficients of $x_1^j x_2^k$ in $\tilde{f}_i(x_1, x_2)$. With this property, a recursive formula is derived as follows.

Let matrix $\mathbf{A}$ be the Jacobian matrix of $f_1, , \ldots, f_{n-2}$ with respect to $x_3, , \ldots, x_n$ evaluated at $\mathbf{x}^0$, i.e.

$$\mathbf{A} \equiv J\mathbf{F}(\mathbf{x}^0) = \begin{bmatrix} a^{(1)}_{0010\ldots0} & \cdots & a^{(1)}_{00\ldots01} \\ \vdots & \vdots & \vdots \\ a^{(n-2)}_{0010\ldots0} & \cdots & a^{(n-2)}_{00\ldots01} \end{bmatrix}$$

We also define $(d_i(j))_{kl}$ as in [48], which is the coefficient of $x_1^k x_2^l$ in $(\psi_i(x_1, x_2))^j$, that is,

$$(\psi_i(x_1, x_2))^j = \sum_{k+l \geq j} (d_i(j))_{kl} x_1^k x_2^l$$

Analogous to the recursive derivation for $(a(k))_{ij}$ in Section 2.2.1, the $(d_i(j))_{kl}$ can be recursively defined as

$$(d_i(j))_{kl} = \begin{cases} 1 & \text{if } j = k = l = 0 \\ 0 & \text{if } j = 0 \text{ and } k + l \geq 1 \\ b^{(i)}_{kl} & \text{if } j = 1 \text{ and } k + l \geq 1 \\ \sum_{j-1 \leq p+q \leq k+l-1} (d_i(j-1))_{pq} b^{(i)}_{(k-p)(l-q)} & \text{if } j \geq 2 \text{ and } k + l \geq j \end{cases}$$

The formula for computing $b^{(i)}_{jk}$, where $j + k \geq 1$ and $i = 3, 4, \ldots, n$, is:

For $j + k = 1$

$$\mathbf{A} \begin{bmatrix} b^{(3)}_{10} \\ \vdots \\ b^{(n)}_{10} \end{bmatrix} = \begin{bmatrix} -a^{(1)}_{100\ldots0} \\ \vdots \\ -a^{(n-2)}_{100\ldots0} \end{bmatrix}$$

and

$$\mathbf{A} \begin{bmatrix} b^{(3)}_{01} \\ \vdots \\ b^{(n)}_{01} \end{bmatrix} = \begin{bmatrix} -a^{(1)}_{010\ldots0} \\ \vdots \\ -a^{(n-2)}_{010\ldots0} \end{bmatrix}$$

For $j+k \geq 2$

$$\mathbf{A}\begin{bmatrix} b_{jk}^{(3)} \\ \vdots \\ b_{jk}^{(n)} \end{bmatrix} = \begin{bmatrix} -r_{jk}^{(1)} \\ \vdots \\ -r_{jk}^{(n-2)} \end{bmatrix}$$

where

$$r_{jk}^{(i)} = a_{jk0\ldots0}^{(i)} + \sum_{j_1+j_2=0}^{j+k-1} \sum_{\substack{k_3+\cdots+k_n=j-j_1 \\ l_3+\cdots+l_n=k-j_2}} \underbrace{\sum_{j_3=0}^{k_3+l_3} \cdots \sum_{j_n=0}^{k_n+l_n}}_{j_3+\cdots+j_n \geq 2-j_1-j_2} a_{j_1 j_2 \ldots j_n}^{(i)} (d_3(j_3))_{k_3 l_3} \cdots (d_n(j_n))_{k_n l_n}$$

Example 3.6 Consider the surface $g = 0$ and its offset given in Example 1.1 and the surface point $\mathbf{x}^0 = (x^0, y^0, z^0, u^0, v^0, w^0) = (0, 0, 4, 0, 0, 2)$. First of all, we have to translate $\mathbf{x}^0$ to the origin, and as the result, system (1.4) becomes

$$\begin{aligned} \hat{z}^2 - 2\hat{w}\hat{z} + y^2 - 2vy + x^2 - 2ux + \hat{w}^2 + v^2 + u^2 + 4\hat{z} - 4\hat{w} &= 0 \\ \hat{w}^2 + v^2 + 4u^2 + 4\hat{w} &= 0 \\ -4u\hat{z} + \hat{w}x + 3u\hat{w} + 2x - 10u &= 0 \\ -v\hat{z} + \hat{w}y + 2y - 4v &= 0 \end{aligned}$$

which has the Jacobian matrix $\mathbf{A}$ with respect to $\hat{z}, u, v, \hat{w}$ and evaluated at $(0,0,0,0,0,0)$,

$$\mathbf{A} = \begin{bmatrix} 4 & 0 & 0 & -4 \\ 0 & 0 & 0 & 4 \\ 0 & -10 & 0 & 0 \\ 0 & 0 & -4 & 0 \end{bmatrix}$$

Solving two linear systems, we have coefficients for linear terms

$$\begin{aligned} (b_{10}^{(3)}, b_{10}^{(4)}, b_{10}^{(5)}, b_{10}^{(6)}) &= (0, 1/5, 0, 0) \\ (b_{01}^{(3)}, b_{01}^{(4)}, b_{01}^{(5)}, b_{01}^{(6)}) &= (0, 0, 1/2, 0) \end{aligned}$$

For $j+k=2$, we have

$$\begin{aligned} (-r_{20}^{(1)}, -r_{20}^{(2)}, -r_{20}^{(3)}, -r_{20}^{(4)}) &= (-1, 0, 0, 0) \\ (-r_{11}^{(1)}, -r_{11}^{(2)}, -r_{11}^{(3)}, -r_{11}^{(4)}) &= (0, 0, 0, 0) \\ (-r_{02}^{(1)}, -r_{02}^{(2)}, -r_{02}^{(3)}, -r_{02}^{(4)}) &= (-1, 0, 0, 0) \end{aligned}$$

and have coefficients

$$\begin{aligned}(b_{20}^{(3)}, b_{20}^{(4)}, b_{20}^{(5)}, b_{20}^{(6)}) &= (-1/4, 0, 0, 0)\\ (b_{11}^{(3)}, b_{11}^{(4)}, b_{11}^{(5)}, b_{11}^{(6)}) &= (0, 0, 0, 0)\\ (b_{02}^{(3)}, b_{02}^{(4)}, b_{02}^{(5)}, b_{02}^{(6)}) &= (-1/4, 0, 0, 0)\end{aligned}$$

Thus, $\hat{z} = -(x^2+y^2)/4$ and after the reverse translation $z = 4-(x^2+y^2)/4$ is the degree 2 local explicitization of the surface at $(0,0,4,0,0,2)$. ■

3.4 Local Parametric Approximation

In [14], a Taylor approximant of the intersection curve of two implicit surfaces is constructed which serves as the local parametrization of the intersection curve. We generalize this method to our problem domain.

For a given system of equations (3.1) and a point $\mathbf{x}^0$ on it, we seek a parametrically described solution

$$\Phi(u,v) = (\phi_1(u,v), \phi_2(u,v), \ldots, \phi_n(u,v)) \text{ with } \Phi(0,0) = \mathbf{x}^0 \qquad (3.9)$$

in the neighborhood V of $\mathbf{x}^0$. Solution (3.9) is basically a local parametrization of the surface represented by (3.1) at point $\mathbf{x}^0$. The three coordinate functions

$$x_1 = \phi_1(u,v), \quad x_2 = \phi_2(u,v), \quad x_3 = \phi_3(u,v)$$

define a *local parametrization* of the projected surface (3.2) in (x_1, x_2, x_3)-space.

When the hypersurfaces f_i intersect transversally and are not singular in the vicinity of $\mathbf{x}^0$ such a parametrically defined solution exists. In the neighborhood of $\mathbf{x}^0$, V, the surfaces are hence defined as the solution of

$$\hat{f}_i(u,v) \equiv f_i(\phi_1(u,v), \phi_2(u,v), \ldots, \phi_n(u,v)) = 0, \quad i = 1, 2, \ldots, n-2. \qquad (3.10)$$

The Taylor expansion of $\hat{f}_i(u,v)$ in power of u and v is

$$\hat{f}_i(u,v) = \hat{f}_i(0,0) + u\frac{\partial \hat{f}_i}{\partial u} - v\frac{\partial \hat{f}_i}{\partial v} + \frac{u^2}{2}\frac{\partial^2 \hat{f}_i}{\partial u^2} - uv\frac{\partial^2 \hat{f}_i}{\partial v \partial v} + \frac{v^2}{2}\frac{\partial^2 \hat{f}_i}{\partial v^2} + \cdots$$

$$
\begin{aligned}
&= f_i(\Phi(0,0)) + u\left(\sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j}\frac{\partial \phi_j}{\partial u}\right) + v\left(\sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j}\frac{\partial \phi_j}{\partial v}\right) \\
&\quad + \frac{u^2}{2}\sum_{k=1}^{n}\left\{\left[\left(\sum_{j=1}^{n} \frac{\partial^2 f_i}{\partial x_j \partial x_k}\frac{\partial \phi_j}{\partial u}\right)\frac{\partial \phi_k}{\partial u}\right] + \frac{\partial f_i}{\partial x_k}\frac{\partial^2 \phi_k}{\partial u^2}\right\} + \cdots \\
&= f_i(\Phi(0,0)) + u\nabla f_i(\mathbf{x}^0)\cdot\Phi_u(0,0) + v\nabla f_i(\mathbf{x}^0)\cdot\Phi_v(0,0) \\
&\quad + \frac{u^2}{2}[\nabla f_i(\mathbf{x}^0)\cdot\Phi_{uu}(0,0) + \Phi_u(0,0)\cdot H_{f_i}\cdot\Phi_u(0,0)] \\
&\quad + uv[\nabla f_i(\mathbf{x}^0)\cdot\Phi_{uv}(0,0) + \Phi_v(0,0)\cdot H_{f_i}\cdot\Phi_u(0,0)] \\
&\quad + \frac{v^2}{2}[\nabla f_i(\mathbf{x}^0)\cdot\Phi_{vv}(0,0) + \Phi_v(0,0)\cdot H_{f_i}\cdot\Phi_v(0,0)] \\
&\quad + \cdots
\end{aligned} \tag{3.11}
$$

where H_{f_i}, the Hessian of f_i, and all the derivatives of f_i are evaluated at $\mathbf{x}^0$, and all the derivatives of $\hat{f}_i$ and ϕ_i are evaluated at $(0,0)$.

In equation (3.10), the coefficient of each monomial $u^j v^k$ must vanish, so by requiring the coefficient of $u^j v^k$ be zero for $1 \le j+k \le m$, a *truncated* local parametrization of degree m can be derived. This amounts to solving the following series of linear systems

$$
\nabla f_i(\mathbf{x}^0)\cdot\Phi_{u^j v^k}(0,0) = B_{jk}^{(i)}, \qquad i = 1,2,\ldots,n-2 \tag{3.12}
$$

where $1 \le j+k \le m$, $\Phi_{u^j v^k}(0,0) = \left(\frac{\partial^{j+k}\phi_i}{\partial u^j \partial v^k}(0,0)\right)_{i=1,2,\ldots,n}$, and the $B_{jk}^{(i)}$ are expressions of partial derivatives of f_i and lower degree partial derivatives of ϕ_i, for example,

$$
B_{10}^{(i)} = 0 \tag{3.13}
$$

$$
B_{01}^{(i)} = 0 \tag{3.14}
$$

$$
B_{20}^{(i)} = -\Phi_u(0,0)\cdot H_{f_i}\cdot\Phi_u(0,0) \tag{3.15}
$$

$$
B_{11}^{(i)} = -\Phi_v(0,0)\cdot H_{f_i}\cdot\Phi_u(0,0) \tag{3.16}
$$

$$
B_{02}^{(i)} = -\Phi_v(0,0)\cdot H_{f_i}\cdot\Phi_v(0,0) \tag{3.17}
$$

Note that the $B_{jk}^{(i)}$ can be recursively defined at $\mathbf{x}^0$ is the origin $(0,\ldots,0)$, in analogy to Section 3.3.

The system (3.12) is a $(n-2) \times n$ system. Since we assume that $D\mathbf{F}(\mathbf{x}^0)$ has rank $n-2$, the solution of this system can be written as the linear combination of $\mathbf{t}_1, \mathbf{t}_2, \nabla f_1(\mathbf{x}^0), \ldots, \nabla f_{n-2}(\mathbf{x}^0)$, where $\mathbf{t}_1$ and $\mathbf{t}_2$ form a unit vector base of the tangent space at $\mathbf{x}^0$, that is,

$$\Phi_{u^j v^k}(0,0) = \alpha_{jk}^{(1)}\mathbf{t}_1 + \alpha_{jk}^{(2)}\mathbf{t}_2 + \beta_{jk}^{(1)}\nabla f_1(\mathbf{x}^0) + \cdots + \beta_{jk}^{(n-2)}\nabla f_{n-2}(\mathbf{x}^0) \qquad (3.18)$$

Substituting into (3.12) yields

$$\sum_{l=1}^{n-2} \beta_{jk}^{(l)} \nabla f_i(\mathbf{x}^0) \cdot \nabla f_l(\mathbf{x}^0) = B_{jk}^{(i)}, \quad i = 1, 2, \ldots, n-2. \qquad (3.19)$$

from which the unique solution, $\beta_{jk}^{(1)}, \ldots, \beta_{jk}^{(n-2)}$, can be derived. Hence, the expression (3.18) with $\alpha_{jk}^{(1)}$ and $\alpha_{jk}^{(2)}$ arbitrarily selected is the general solution of the system (3.12). The suitable values for $\alpha_{jk}^{(1)}$ and $\alpha_{jk}^{(2)}$ are chosen as follows:

1. For systems (3.13) and (3.14), we have

$$\beta_{jk}^{(1)} = \beta_{jk}^{(2)} = \ldots = \beta_{jk}^{(n-2)} = 0$$

   We assign $\alpha_{jk}^{(1)} = 1$ and $\alpha_{jk}^{(2)} = 0$ for (3.13) and $\alpha_{jk}^{(1)} = 0$ and $\alpha_{jk}^{(2)} = 1$ for (3.14) so that the isoparametric curves $\Phi((u,0))$ and $\Phi((0,v))$ intersect transversally at point $\mathbf{x}^0$.

2. For systems (3.15), (3.16) and (3.17), we assign $\alpha_{jk}^{(1)} = \alpha_{jk}^{(2)} = 0$.

Note that the three coordinate functions

$$x_1 = \phi_1(u,v), \quad x_2 = \phi_2(u,v), \quad x_3 = \phi_3(u,v)$$

obtained in this way define a local parametrization of the surface (3.2) in (x_1, x_2, x_3)-space.

To compute the solution of system (3.12) with a numerically stable method, we consider singular value decomposition, see e.g., [29]. The matrix $(D\mathbf{F}(\mathbf{x}^0))^T$ is factored as

$$(D\mathbf{F}(\mathbf{x}^0))^T = U\Sigma V^T$$

where $U = [\mathbf{u}_1, \ldots, \mathbf{u}_n] \in \mathbf{R}^{n\times n}$ and $V = [\mathbf{v}_1, \ldots, \mathbf{v}_{n-2}] \in \mathbf{R}^{(n-2)\times(n-2)}$ are orthogonal matrices, and $\Sigma \in \mathbf{R}^{n\times(n-2)}$ is a diagonal matrix. With direct substitution of the factorization, system (3.12) becomes

$$V\Sigma^T U^T \Phi_{u^j v^k}(0,0) = B$$

where $B = [B_{jk}^{(1)}, \ldots, B_{jk}^{(n-2)}]^T$. Its solution can be generally written as

$$\Phi_{u^j v^k}(0,0) = \gamma_1 \mathbf{u}_1 + \gamma_2 \mathbf{u}_2 + \cdots + \gamma_n \mathbf{u}_n$$

and because the rank of the differential matrix is $n-2$, $\Sigma_{ii} \neq 0$ for $i = 1, 2, \ldots, n-2$, and thus the $\gamma_1, \ldots, \gamma_{n-2}$ are uniquely determined by

$$\gamma_i = (\mathbf{v}_i^T B)/\Sigma_{ii}$$

and γ_{n-1} and γ_n are arbitrary. Here $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{n-2}$ span the range of $(D\mathbf{F}(\mathbf{x}^0))^T$, hence span the same space as the gradients. Note also that the null space of $D\mathbf{F}(\mathbf{x}^0)$ spanned by $\mathbf{u}_{n-1}$ and $\mathbf{u}_n$. Thus we obtain $\Phi_u(0,0) = \mathbf{u}_{n-1}$ and $\Phi_v(0,0) = \mathbf{u}_n$. For $\Phi_{u^2}(0,0)$, $\Phi_{uv}(0,0)$, and $\Phi_{v^2}(0,0)$, we let $\gamma_{n-1} = \gamma_n = 0$.

Example 3.7 Consider the surface $g = 0$ and its offset given in Example 1.1 and the surface point $\mathbf{x}^0 = (x^0, y^0, z^0, u^0, v^0, w^0) = (0.0, 4, 0, 0, 2)$. It is clear that $\Phi(0,0) = (0, 0, 4, 0, 0, 2)$, $\Phi_s(0,0) = \mathbf{t}_1$ and $\Phi_t(0,0) = \mathbf{t}_2$, where $\mathbf{t}_1 = (5/\sqrt{26}, 0, 0, 1/\sqrt{26}, 0, 0)$ and $\mathbf{t}_2 = (0, 2/\sqrt{5}, 0, 0, 1/\sqrt{5}, 0)$. By applying formula (3.19), we have

$$\begin{aligned}
(\beta_{20}^{(1)}, \beta_{20}^{(2)}, \beta_{20}^{(3)}, \beta_{20}^{(4)}) &= (-5/52, -3/26, 0, 0) \\
(\beta_{11}^{(1)}, \beta_{11}^{(2)}, \beta_{11}^{(3)}, \beta_{11}^{(4)}) &= (0, 0, 0, 0) \\
(\beta_{02}^{(1)}, \beta_{02}^{(2)}, \beta_{02}^{(3)}, \beta_{02}^{(4)}) &= (-1/20, -3/40, 0, 0)
\end{aligned}$$

Thus, from formula (3.18), it follows that

$$\begin{aligned}
\Phi_{s^2}(0,0) &= (0, 0, -5/13, 0, 0, -1/13) \\
\Phi_{st}(0,0) &= (0, 0, 0, 0, 0, 0) \\
\Phi_{t^2}(0,0) &= (0, 0, -1/5, 0, 0, -1/10)
\end{aligned}$$

Consequently, we have

$$\begin{aligned}\phi_1(s,t) &= \tfrac{5}{\sqrt{26}}s \\ \phi_2(s,t) &= \tfrac{2}{\sqrt{5}}t \\ \phi_3(s,t) &= 4 - \tfrac{5}{26}s^2 - \tfrac{1}{10}t^2\end{aligned}$$

as the local parametric approximant of degree 2 in (x_1, x_2, x_3)-space. ■

4. PIECEWISE APPROXIMATIONS OF 2-D SURFACES

Implicit surfaces have recently become more important in CAGD and solid modeling. In part, implicit surfaces have specific advantages over the traditional parametric surfaces. For example, many complex objects can be modeled more easily using implicit surfaces, and certain geometric operations, e.g., membership classification problem, can be handled straightforwardly when implicit surface representations are available. Moreover, using implicit surfaces, a number of sophisticated techniques have been proposed, e.g., the substitution blending surfaces of [34, 35, 36, 37].

As the role of implicit surfaces increases in importance in solid modeling and CAGD, rendering implicit surfaces efficiently becomes a crucial support in surface design. In computer graphics, many surface rendering algorithms rely on piecewise linear approximations (PLAs) of a surface since a PLA allows one to take advantage of hardware capabilities and reduces the cost of expensive ray casting in the rendering process. However, while the PLA of parametric surfaces has been extensively studied and utilized as a tool for the evaluation of surface intersections [10, 16, 38, 41, 42] and for rendering, it seems that much less attention has been paid in the literature to the PLA of implicitly defined surfaces. Recently, Bloomenthal [17] has proposed an algorithm for computing the PLA of an implicit surface based on space subdivision using octrees. In [9, 7, 8, 6], a simplicial continuation algorithm is presented for obtaining a PLA to a component of an implicit surface. Both methods fundamentally rely on vertex evaluation. Rheinboldt [56, 57] has presented an algorithm that maps a triangulation of $\mathbf{R}^p$ to a p-manifold, where $p \geq 1$, and hence induces a triangulation on the p-manifold.

Offset and Voronoi surfaces of implicit surfaces can be formulated as the projection to $\mathbf{R}^3$ of 2-surfaces that are implicitly defined in six and ten dimensional spaces respectively. Algorithms using space subdivision or simplicial continuation can be generalized to compute the PLA of the 2-surface in high-dimensional space. Thus, to compute the PLA of the offset of an implicit surface, we could compute the PLA of the 2-surface in 6-space, and then project it into 3-D subspaces. However, as the formulation dimension increases, the complexity of computing the PLA increases exponentially. To reduce the complexity, we propose an algorithm that determines the PLA of the projection to $\mathbf{R}^3$ of the 2-surface defined in high-dimensional space, with all major computations performed in 3-space.

In Section 1 we briefly describe a method based on the simplicial continuation method due to Allgower and Gnutzmann [7], and explain some difficulties when applied to offset, Voronoi and blending surfaces. In Section 2 we sketch the proposed algorithm in pseudo code, and then in Section 3 we describe the computations in detail.

4.1 The PLA of Offset, Voronoi and Blending Surfaces

As described in Section 1.1.3, the offset of an implicit surface can be viewed as the projection to $\mathbf{R}^3$ of a 2-surface in $\mathbf{R}^6$, and the Voronoi surface of two implicit surfaces as the projection to $\mathbf{R}^3$ of a 2-surface in $\mathbf{R}^{10}$ as shown in (1.3) and (1.5) respectively. The 2-surfaces are represented by the following system of $n-2$ equations in n variables,

$$\begin{aligned} f_1(x_1, x_2, \ldots, x_n) &= 0 \\ f_2(x_1, x_2, \ldots, x_n) &= 0 \\ &\vdots \\ f_{n-2}(x_1, x_2, \ldots, x_n) &= 0 \end{aligned} \tag{4.1}$$

abbreviated in matrix form as $\mathbf{F}(\mathbf{x}) = \mathbf{0}$. We suppose that the target surface we are interested in is in (x_1, x_2, x_3)-space. A closed-form $f(x_1, x_2, x_3) = 0$ can be obtained

in principle by eliminating the last $n-3$ variables, but this is often not possible in practice. It is also impractical to compute the PLA of the surface in n-space, using generalizations of space decomposition or simplicial continuation, because the complexity of the computation increases exponentially with the dimension of the space. One way to reduce the complexity might be by computing the PLAs of the desired surface, e.g., offset, and its basis surface in parallel in such a way that the major computation is performed in the orthogonal 3-D subspaces. We have looked into such an approach for computing the PLA of an offset surface. Let system (1.3) define the offset in $\mathbf{R}^6$. For a pair of corresponding points $\mathbf{x}_1$ and $\mathbf{x}_2$ on the offset and its basis surface respectively, we determine two 3-simplices, one containing $\mathbf{x}_1$ in (x, y, z)-space and the other containing $\mathbf{x}_2$ in (u, v, w)-space. Then two sequences of transversal simplices are constructed in parallel. Two sequences of simplices are constructed that are coordinated by the following computations and considerations:

Let σ_1 and σ_2 be two simplices that we are currently processing, $\mathbf{x}_1 \in \sigma_1$ be a point on the offset surface and $\mathbf{x}_2 \in \sigma_2$ be the footpoint of $\mathbf{x}_1$ on the basis surface.

1. Determine the simplex $\sigma \subset \mathbf{R}^6$ such that $(\mathbf{x}_1, \mathbf{x}_2) \in \sigma$ and σ projects to σ_1 and σ_2.

2. Compute the affine map $\mathbf{H}$ that interpolates $\mathbf{F}(\mathbf{x})$ at all the vertices of σ.

3. Project the affine map $\mathbf{H}$ to $\mathbf{H}_1$ and $\mathbf{H}_2$ in the two orthogonal subspaces.

4. Compute the intersections of $\mathbf{H}_i$ with edges of σ_i, for $i = 1, 2$.

5. Pair the intersections with σ_1 and σ_2 to obtain points in 6-space and refine them iteratively back to zeros of $\mathbf{F}(\mathbf{x}) = \mathbf{0}$. The points will be the new $(\mathbf{x}_1, \mathbf{x}_2)$.

6. Based on $\mathbf{x}_1$ and $\mathbf{x}_2$, we determine how to pivot the current pair of simplices. The resulting simplices are the new σ_1 and σ_2.

7. Go back to Step 1.

We have investigated such an approach and found it unattractive for the following reasons:

- For given σ_1 containing $\mathbf{x}_1$ in (x, y, z)-space and σ_2 containing $\mathbf{x}_2$ in (u, v, w)-space, there are 4!4! possible simplices σ in 6-space that project to the same pair σ_1 and σ_2. Hence Step 1 requires considerable computation for generating σ and checking if $(\mathbf{x}_1, \mathbf{x}_2) \in \sigma$.

- For two transversal simplices σ_1 and σ_2, the simplex σ found in Step 1 might not be transversal. Suppose $(\tilde{\sigma}_1, \tilde{\sigma}_2)$ is the pair of simplices in the previous step, corresponding to the simplex $\tilde{\sigma}$ in 6-space. That is, σ_1 and σ_2 are the result of pivoting $\tilde{\sigma}_1$ and $\tilde{\sigma}_2$ in 3-space, respectively, with respect to shared transversal faces. Note that the pivoting of $\tilde{\sigma}_1$ and $\tilde{\sigma}_2$ corresponds to a sequence of pivotings of $\tilde{\sigma}$ which is not necessarily done with respect to transversal faces. Therefore, σ might be not transversal. Now $(\mathbf{x}_1, \mathbf{x}_2) \in \sigma$, and so it is easily seen that in such a case the surface penetrates σ although all vertices of σ have the same signs on vertex evaluation. In this case, the computed affine map $\mathbf{H}$ is useless for continuing the computation.

- The two projected affine maps $\mathbf{H}_1$ and $\mathbf{H}_2$ are generally not the affine maps of σ_1 and σ_2 respectively. Hence the projected affine maps for adjacent 3-simplices are often not continuous at the shared face.

- The intersections of $\mathbf{H}_i$ with σ_i, for $i = 1, 2$, are often not in correspondence, and then it is not possible to define a meaningful point pairing.

- When applied to offsets of parametric surfaces, Steps 4, 5, and 6 require substantial modifications.

Using 3-simplices is attractive because with it vertex evaluation is unambiguous and there is a simple pivoting scheme. However, adaptive subdivision resulting

again in 3-simplices is more complicated. So, if vertex evaluation is not necessary, using cubes to subdivide space might be a better choice since adaptive subdivision is very simple.

The parametric approximant $\Phi(s,t)$, discussed in Section 3.4 could be used in place of an affine map. However, the coordination of the sequences of cubes is complicated, especially when computing Voronoi and blending surfaces where more than two sequences of cubes have to be put into correspondence.

4.2 Proposed Approach

Instead of computing a PLA of the 2-surface in n-space, or its projection into several subspaces, we will compute the PLA of one projection only. In the case of offset surfaces, this means that we approximate the offset in 3-space without explicitly approximating the basis surface as well, or the 2-surface in 6-space corresponding to both the offset and its basis surface. We will use a method similar to simplicial *continuation in* [8], but based on local parametric approximation. The approach is well suited to those 2-surfaces that have been defined in $\mathbf{R}^n$, but whose projection to 3-space is ultimately of interest.

Let $\mathbf{F}(\mathbf{x}) = \mathbf{0}$ be the surface definition in $\mathbf{R}^n$, S_f its projection into (x_1, x_2, x_3)-space. Given a regular surface point $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2) \in \mathbf{R}^n$, where $\mathbf{x}_1 \in \mathbf{R}^3$ and $\mathbf{x}_2 \in \mathbf{R}^{n-3}$, and a cube in $\mathbf{R}^3$ containing $\mathbf{x}_1$, a sequence of consecutive cubes intersecting the surface S_f is generated in a fashion that spirals outward from $\mathbf{x}_1$. For each cube, a degree two parametric approximant $\Phi(s,t) = (\phi_1(s,t), \phi_2(s,t), \ldots, \phi_n(s,t))$ of $\mathbf{F}(\mathbf{x}) = \mathbf{0}$ is derived, and $\Phi_1(s,t) = (\phi_1(s,t), \phi_2(s,t), \phi_3(s,t))$ serves as the surface approximation of the target surface in (x_1, x_2, x_3)-space. The parametric approximant $\Phi(s,t)$ plays an essential role in this approach. First of all, the intersections of $\Phi_1(s,t)$ with the edges of the cube are used as estimates that are refined to surface intersections along edges of the cube. Secondly, the parametric approximant provides a way of recovering a point in $\mathbf{R}^n$ while doing computations

in $\mathbf{R}^3$. Thirdly, when extending the surface approximation into a neighboring cube, $\Phi(s,t)$ provides a mechanism for obtaining an estimate to a surface point $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2) \in \mathbf{R}^n$ such that $\mathbf{x}_1$ lies in the neighboring cube. With this parametric approximant, we are able to *march* on the surface in (x_1, x_2, x_3)-space although the surface is formally defined in $\mathbf{R}^n$, $n > 3$. This approach is of course also applicable to offsets, Voronoi surfaces and blends where some of the basis surfaces are in parametric form. For the computation of parametric approximation, see Section 3.4. We give a high-level description of the approximation algorithm.

Algorithm 4.1

Input

$\mathbf{F}(\mathbf{x}) = \mathbf{0}$

$\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2) \in \mathbf{R}^n$: a regular surface point where $\mathbf{x}_1 \in (x_1, x_2, x_3)$-space

B: a cube with edge size δ containing $\mathbf{x}_1$

D: a compact domain $(a_1 : a_2, b_1 : b_2, c_1 : c_2)$ in (x_1, x_2, x_3)-space

Output

L: List of transversal cubes and their surface intersections where the cubes are contained in D or intersect the boundary of D.

begin

(1) $W = \emptyset$; $L = \emptyset$;

repeat

(2) Compute the degree 2 parametric approximant $\Phi(s,t)$ of $\mathbf{F}(\mathbf{x}) = \mathbf{0}$ at $\mathbf{x}$;

(3) Compute the intersections of $\Phi_1(s,t)$ with edges of B;

(4) Refine the intersections back to the projected surface S_f along the edges of B;

(5) $face(B) = \{ [F_i, < (e_{i1}, p_{i1}), (\bar{e}_{i1}, (s_{i1}, t_{i1})) >, < (e_{i2}, p_{i2}), (\bar{e}_{i2}, (s_{i2}, t_{i2})) >] \mid$

F_i is a face of B, e_{i1} and e_{i2} are edges of face F_i.

S_f intersects e_{ij} at p_{ij} where p_{ij} is the point refined

from $\Phi_1(s_{ij}, t_{ij})$, and $\Phi_1(s_{ij}, t_{ij})$ is on $\bar{e}_{ij}$, $j = 1, 2$ };

(6) $L = L \cup \{< B, \Phi(s,t), face(B) >\}$;

(7) Remove those faces in $face(B)$ that are outside D;

(8) **for** $\bar{B} \in W$ **begin**

(8-1) **if** B and $\bar{B}$ have a common face F
and common surface intersections p_1 and p_2
then begin

(8-2) $face(B) = face(B) - \{[F, < (e_1, p_1), (e_1, (s_1, t_1)) >,$
$< (e_2, p_2), (\hat{e}_2, (s_2, t_2)) >]\}$;

(8-3) $face(\bar{B}) = face(\bar{B}) - \{[F, < (e_1, p_1), (\bar{e}_1, (\bar{s}_1, \bar{t}_1)) >,$
$< (e_2, p_2), (\bar{e}_2, (\bar{s}_2, \bar{t}_2)) >]\}$;

(8-4) **if** $face(\bar{B}) = \emptyset$ **then** $W = W - \{\bar{B}\}$;
end /* if */
end /* for */

(9) **if** $face(B) \neq \emptyset$ **then begin**

(10) $W = W \cup \{B\}$;

(11) $\check{B} = B$;
end

(12) **else** Select a cube $\check{B}$ from W;

(13) Select $[F, < (e_1, p_1), (\bar{e}_1, (s_1, t_1)) >, < (e_2, p_2), (\bar{e}_2, (s_2, t_2)) >] \in face(\check{B})$;

(14) Determine (u_0, v_0) from (s_1, t_1) and (s_2, t_2) such that $\Phi(u_0, v_0)$ can be refined to a surface point $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$ where $\mathbf{x}_1$ is in the cube B that shares F with $\check{B}$;

until $W = \emptyset$

end

4.3 Detailed Computations

4.3.1 Intersecting A Parametric Approximant with Edges of a Cube

A crucial step in the Algorithm 4.1 is the computation of points at which the parametric approximant intersects the cube's edges. This computation is more complicated than intersection with an implicit approximant. With an implicit surface approximant, vertex evaluation can be used to determine which edges intersect the surface and then the intersection point on each edge can be estimated by subdivision or interpolation [17, 45, 8].

When tracing the intersections of $\Phi_1(s,t)$ with B, at least one closed loop of curve segments will be formed and each such loop corresponds to a closed loop in the (s,t) parameter space. Since $\Phi_1(s,t)$ approximates the surface locally within a cube B and the intersection points along the edges of B are only used as approximates to surface intersections of S_f with edges of B, only the "nearest" intersections with $\Phi_1(s,t)$ are of interest. More precisely, only those intersections whose parameter values lie on the innermost loop containing $(0,0)$ are considered. In the following, when we mention surface intersections with edges of a cube, only surface-edge intersections of this kind are referred to. It is easy to see that $\Phi_1(s,t)$ may intersect the edges of B in $3,4,5$, or 6 points, as shown in Figure 4.1. As degenerate cases, $\Phi_1(s,t)$ can penetrate a face of B without intersecting any edge or intersect only one edge of B in two points, see Figure 4.2. We formulate the problem of finding the intersections as follows:

Problem 4.1 For given $\Phi_1(s,t) = (\phi_1(s,t), \phi_2(s,t), \phi_3(s,t))$ of degree 2 and a cube B containing $\Phi_1(0,0)$, compute the edge intersection points of $\Phi_1(s,t)$ with B.

As a subproblem of (4.1), we explain how to compute the intersection of $\Phi_1(s,t)$ with an edge of B. The line L containing an edge $\overline{uv}$ of the cube B is the intersection of two face planes $P_1(x_1,x_2,x_3) = 0$ and $P_2(x_1,x_2,x_3) = 0$. Thus, substitution

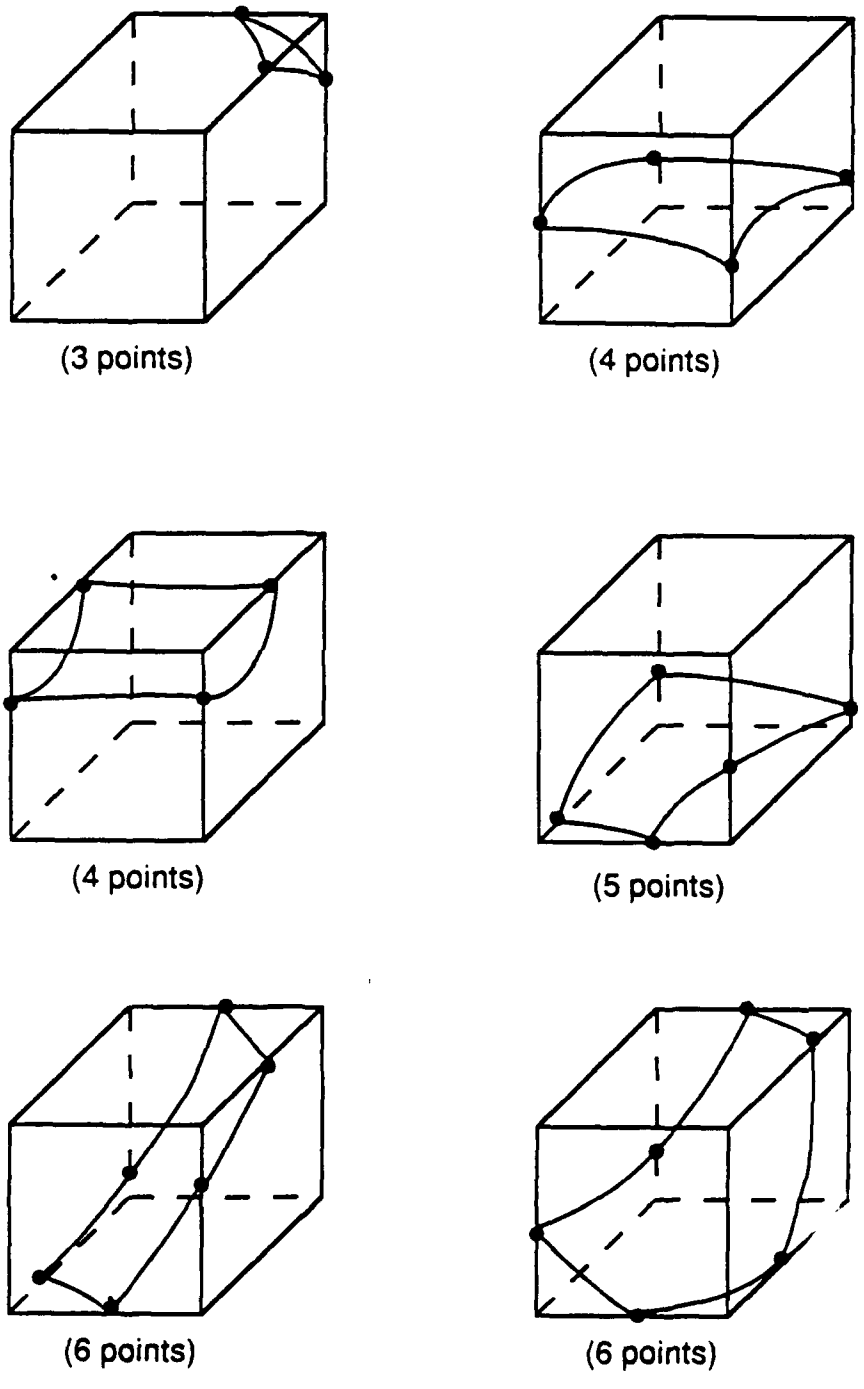


Figure 4.1 Intersecting $\Phi_1(s,t)$ with a cube Γ (Regular cases)

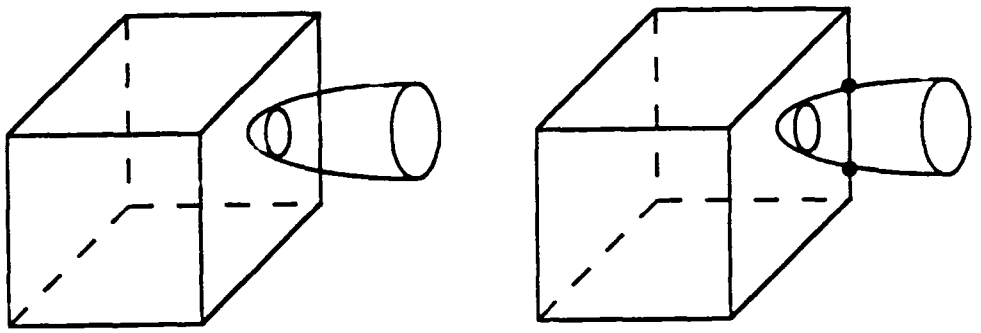


Figure 4.2 Intersecting $\Phi_1(s,t)$ with a cube B (Degenerate cases)

of $\Phi_1(s,t)$ into P_1 and P_2 yields the two equations

$$\begin{aligned} p_1(s,t) &= 0 \\ p_2(s,t) &= 0 \end{aligned} \tag{4.2}$$

which represent the intersections of $\Phi_1(s,t)$ with L in parameter space. For quadratic approximants, $p_1(s,t)$ and $p_2(s,t)$ are of degree 2. System (4.2) can be solved either algebraically or numerically.

Using a resultant method, we obtain the resultant of p_1 and p_2 as a degree 4 polynomial in, e.g., s. This univariate polynomial is then solved by a root finding algorithm. Substituting each root $\hat{s}$ into $p_1(s,t) = 0$ and $p_2(s,t) = 0$ results in a polynomial in t which yields solutions $\hat{t}$. The computed solutions $(\hat{s},\hat{t})$ are checked to see if $\Phi_1(\hat{s},\hat{t})$ is on $\overline{uv}$. Since there might be more than one such $(\hat{s},\hat{t})$, it is not trivial to determine which $(\hat{s},\hat{t})$ is nearest. The computation has to be done for all edges of B in order to find the desired intersections of $\Phi_1(s,t)$ along edges.

To apply Newton iteration to system (4.2), an initial point is needed. Possible initial points would be (s_0,t_0) values arising as the intersection of the tangent plane

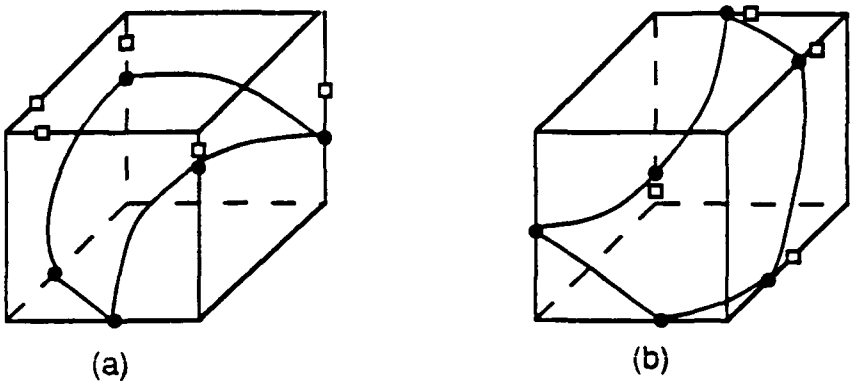


Figure 4.3 Intersecting a tangent plane with the cube B

of $\Phi_1(0,0)$ with the edges. However, these initial points represented as (s_0, t_0) in the parameter space might not be close enough to the true intersections if the cube is large compared to the radius of curvature of Φ_1. See Figure 4.3(a). Moreover, we might have fewer initial points than the number of intersections with Φ_1, and it is unclear how to ascertain that all intersections have been found. See also Figure 4.3(b). Notice that

1. If the isoparametric curves $\Phi_1(s,0)$ or $\Phi_1(0,t)$ intersect a face F of cube B then $\Phi_1(s,t)$ intersects edges of F at least at 2 points.

2. All intersections of $\Phi_1(s,t)$ to edges of B form a closed loop.

With the above observations, we derive the following algorithm, which is reliable and fairly efficient.

Algorithm 4.2 (For Problem 4.1)

Input

B: a cube

$\Phi_1(s,t)$: a degree 2 parametric surface with $\Phi_1(0,0)$ inside B.

begin

(1) $C = \emptyset$;

(2) Find an intersection $\Phi_1(s_1,t_1)$ with an edge e_1 of B,

where $e_1 = F_0 \cap F_1$, F_0 and F_1 are faces of B;

(3) $LF = F_0$; $Le = e_1$; $Ls = s_1$; $Lt = t_1$;

(4) **repeat**

(5) Find the second intersection $\Phi_1(s_2,t_2)$ on an edge e_2 of F_1,

where $e_2 = F_1 \cap F_2$, F_2 is a face of B;

(6) $C = C \cup \{< ((F_0,F_1),e_1,(s_1,t_1)),((F_1,F_2),e_2,(s_2,t_2)) >\}$;

(7) $F_0 = F_1$; $F_1 = F_2$; $e_1 = e_2$; $s_1 = s_2$; $t_1 = t_2$;

until $F_0 = LF$ and $(s_1,t_1) = (Ls,Lt)$;

end

For step (2) of Algorithm 4.2, we do the following:

Algorithm 4.3

1. Find a face F_0 of B that $\Phi_1(s,0)$ intersects, say at $(\hat{s},0)$:
   (a) Substitute $\Phi_1(s,0)$ into the equation of plane F_0 and solve the resulting degree 2 univariate polynomial in s.
   (b) Take the real root $\hat{s}$, if there is one, that is closest to zero and determine if $\Phi_1(\hat{s},0)$ is inside F_0. If so, we have found the intersection.
2. (a) Substitute $\Phi_1(s,t)$ into the equation of plane F_0 obtaining a plane curve $p(s,t) = 0$.
   (b) Trace $p(s,t) = 0$ starting at $(\hat{s},0)$ until one edge $e_1 = \overline{uv}$ of F_0 is crossed over at $(\bar{s},\bar{t})$.

(c) Apply Newton iteration to System (4.2) and refine $(\bar{s}, \bar{t})$ to a solution (s_1, t_1) of (4.2). $\Phi_1(s_1, t_1)$ is the surface point on e_1.

For step (4) of Algorithm 4.2, a computation similar to step (2) of Algorithm 4.2 can be used. Note that the initial tracing direction can be determined easily, namely the one that is pointing to the inside of the face F_1. For tracing plane curves, the algorithm and implementation of [14] is used.

4.3.2 Intersecting A Projected Surface with Edges of a Cube

The intersections of the projected surface S_f with the edges of a cube B not only determine the faces that are transversal to S_f but also provide a PLA of S_f within the cube. Let $P_1(x_1, x_2, x_3) = 0$ and $P_2(x_1, x_2, x_3) = 0$ be the two face planes whose intersection contains an edge e of B. A surface point (x_1, x_2, x_3) on e, if there is one, together with its footpoint $(x_4, \ldots, x_n)$ on the basis surface(s) satisfies the following system of n equations in n variables

$$\begin{aligned} f_1(x_1, x_2, \ldots, x_n) &= 0 \\ f_2(x_1, x_2, \ldots, x_n) &= 0 \\ &\vdots \\ f_{n-2}(x_1, x_2, \ldots, x_n) &= 0 \\ P_1(x_1, x_2, x_3) &= 0 \\ P_2(x_1, x_2, x_3) &= 0 \end{aligned} \tag{4.3}$$

We compute the surface points of S_f on edges of B by applying Newton iteration to (4.3) using the intersections of $\Phi_1(s, t)$ with edges of B as initial points. That is, when $\Phi_1(s_1, t_1)$ is a point on an edge $\tilde{e}_1$ of B, $\Phi(s_1, t_1) \in \mathbf{R}^n$ serves as an approximation of a zero of (4.3) whose natural projection to (x_1, x_2, x_3)-space, say p, is the refined point on an edge e_1 of B. In short, we say $\Phi_1(s_1, t_1)$ on $\tilde{e}_1$ is refined to p on e_1. An approximation $\Phi_1(s_1, t_1)$ can be refined to a surface point on the same edge or to a surface point on an adjacent edge. Thus, the approximation can be refined to one point or two points as shown in Figures 4.4 and 4.5 respectively. In

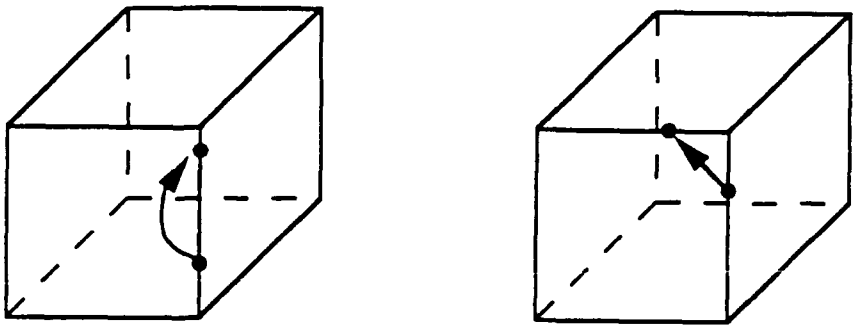


Figure 4.4 An approximate point is refined to a surface point

the latter case, an additional transversal face F to S_f is introduced. Moreover, two approximates $\Phi_1(s_1, t_1)$ and $\Phi_1(s_2, t_2)$ might be refined to the same surface point; see Figure 4.6. In this case, the face F is not transversal to S_f. We describe later how to predict to which edge an approximant is refined. Nevertheless, when an approximant is refined to an edge e using (4.3), P_1 and P_2 are the plane equations for two faces whose intersection is e.

When an approximate point $\Phi_1(s_1, t_1)$ is on an edge e_1 of B, it is assumed that the projected surface S_f will intersect e_1 or its adjacent edges e_2, e_3, e_4, e_5, and further subdivision is necessary if this is not the case. When Newton iteration fails to refine $\Phi_1(s_1, t_1)$ to a point on $e_i, i = 1, \ldots, 5$, we subdivide the cube B into eight equal-sized cubes in order to acquire better approximations. The subdivision will be discussed in Section 4.3.6.

Algorithm 4.4 accepts as input a cube B, its list of k transversal faces to $\Phi_1(s, t)$, and the associated intersection points on each face. The list of transversal faces is

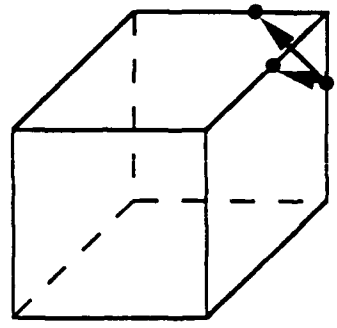


Figure 4.5 An approximate point is refined to two surface points

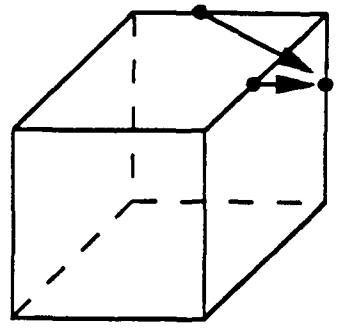


Figure 4.6 Two approximate points are refined to one surface point

in the form

$$\{ [F_i, < \tilde{e}_{i1}, (s_{i1}, t_{i1}) >, < \tilde{e}_{i2}, (s_{i2}, t_{i2}) >] \mid i = 1, \ldots, k\}$$

where F_i is a transversal face, and $\tilde{e}_{ij}$ is the edge of B on which $\Phi_1(s_{ij}, t_{ij})$ lies, for $j = 1, 2$. The algorithm considers each transversal face F_i, refines $\Phi_1(s_{i1}, t_{i1})$ and $\Phi_1(s_{i2}, t_{i2})$ and determines if F_i is transversal to S_f. When the two approximates are refined to two different surface points on edges of F_i, the face F_i is transversal to S_f. If the two approximates are refined to the same surface point then F_i is not transversal to S_f and should be excluded. Finally, if the refined point of $\Phi_1(s_{i1}, t_{i1})$ is different from that of $\Phi_1(s_{(i-1)2}, t_{(i-1)2})$ then the face containing the two refined points is transversal to S_f and should be added to the transversal-face list of S_f. Note that $\Phi_1(s_{i1}, t_{i1})$ does not have to be refined if, on the previous face, $\Phi_1(s_{(i-1)2}, t_{(i-1)2})$ is refined to a point lying on $\tilde{e}_{(i-1)2}$. If this is not the case, we see if $\Phi_1(s_{i1}, t_{i1})$ can be refined to the edge e of F_i that is perpendicular to $\tilde{e}_{i1}$ and $e_{(i-1)2}$. If so, the face F containing e and $e_{(i-1)2}$, rather than F_i, is transversal to S_f; otherwise $\Phi_1(s_{i1}, t_{i1})$ must be refined to $e_{(i-1)2}$; see Figure 4.7. Moreover, once $\Phi_1(s_{i1}, t_{i1})$ is refined to a surface point on an edge of a face F, $\Phi_1(s_{i2}, t_{i2})$ can only be refined to a point on an edge of F.

We describe the algorithm in pseudo code as follows.

Algorithm 4.4

Input

B: a cube

$\{ [F_i, < \tilde{e}_{i1}, (s_{i1}, t_{i1}) >, < \tilde{e}_{i2}, (s_{i2}, t_{i2}) >] \mid i = 1, \ldots, k\}$: list of transversal faces to $\Phi_1(s, t)$ and intersections on edges.

Output

$face(B)$: list of transversal faces to S_f and intersections in the form of

$\{ [F_i, < (e_{i1}, p_{i1}), (\tilde{e}_{i1}, (s_{i1}, t_{i1})) >, < (e_{i2}, p_{i2}), (\tilde{e}_{i2}, (s_{i2}, t_{i2})) >] \mid i = 1, \ldots, l\}$

begin

(1) $face(B) = \emptyset$;

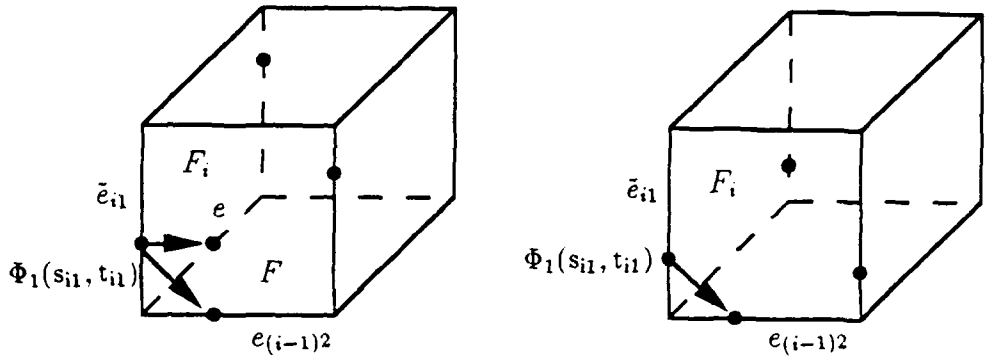


Figure 4.7 The refinement of $\Phi_1(s_{i1}, t_{i1})$

(2) $i = 0$;

repeat

(3) $i = i + 1$;

(4) refine $\Phi_1(s_{i1}, t_{i1})$ to a point p_{i1} on an edge e_{i1} of B;

(5) refine $\Phi_1(s_{i2}, t_{i2})$ to a point p_{i2} on an edge e_{i2} of B;

until $p_{i1} \neq p_{i2}$

Let F be the face containing e_{i1} and e_{i2}

(6) $face(B) = face(B) \cup \{ [F, < (e_{i1}, p_{i1}), (\tilde{e}_{i1}, (s_{i1}, t_{i1})) >,$
$< (e_{i2}, p_{i2}), (\tilde{e}_{i2}, (s_{i2}, t_{i2})) >] \}$;

(7) **do while** $i + 1 \leq k$ **begin**

(8) $i = i + 1$;

(9) **if** $e_{(i-1)2} = \tilde{e}_{(i-1)2}$ **or** $(s_{(i-1)2} \neq s_{i1}$ **and** $t_{(i-1)2} \neq t_{i1})$

then begin /* $\Phi_1(s_{i1}, t_{i1})$ is refined to $p_{(i-1)2}$ */

(10) refine $\Phi_1(s_{i2}, t_{i2})$ to a point p_{i2} on an edge e_{i2} of B;

(11) $face(B) = face(B) \cup \{ [F, < (e_{(i-1)2}, p_{(i-1)2}), (\tilde{e}_{i1}, (s_{i1}, t_{i1})) >,$
$< (e_{i2}, p_{i2}), (\tilde{e}_{i2}, (s_{i2}, t_{i2})) >] \}$;

(12) **end**

else begin

(13) refine $\Phi_1(s_{i1}, t_{i1})$ to a point p_{i1} on an edge e_{i1} of B;

(14) **if** $e_{i1} = F_i \cap F_e$, where $F_e \cap F_{i-1} = e_{(i-1)2}$,

then begin /* $\Phi_1(s_{i1}, t_{i1})$ is refined to 2 different points */

(15) $face(B) = face(B) \cup$

$\{ [F_e, < (e_{(i-1)2}, p_{(i-1)2}), (\tilde{e}_{(i-1)2}, (s_{(i-1)2}, t_{(i-1)2})) >,$

$< (e_{i1}, p_{i1}), (\tilde{e}_{i1}, (s_{i1}, t_{i1})) >] \};$

(16) refine $\Phi_1(s_{i2}, t_{i2})$ to a point p_{i2} on an edge e_{i2} of B;

(17) $face(B) = face(B) \cup \{ [F_i, < (e_{i1}, p_{i1}), (\tilde{e}_{i1}, (s_{i1}, t_{i1})) >,$

$< (e_{i2}, p_{i2}), (\tilde{e}_{i2}, (s_{i2}, t_{i2})) >] \};$

end

else begin

(18) refine $\Phi_1(s_{i2}, t_{i2})$ to a point p_{i2} on an edge e_{i2} of B;

(19) **if** $p_{i1} \neq p_{i2}$ **then begin**

(20) let F be the face containing e_{i1} and e_{i2};

(20) $face(B) = face(B) \cup \{ [F, < (e_{i1}, p_{i1}), (\tilde{e}_{i1}, (s_{i1}, t_{i1})) >,$

$< (e_{i2}, p_{i2}), (\tilde{e}_{i2}, (s_{i2}, t_{i2})) >] \};$

end /*then*/

end /*else*/

end /*else*/

end /*do*/

end

4.3.3 Strategies for Stepping

After transversal faces of a cube $\tilde{B}$ have been found, we track the surface S_f by propagating cubes along transversal edges, taking care that no cube is processed twice. When considering an adjacent transversal cube B, a surface point $(\mathbf{x}_1, \mathbf{x}_2)$ such that $\mathbf{x}_1 \in B$ is required for computing the local approximant $\Phi(s, t)$ from which $\Phi_1(s, t)$ serves as the approximation of S_f inside B. To find such a surface

point $(\mathbf{x}_1, \mathbf{x}_2)$, one possibility is to extend the surface approximant $\tilde{\Phi}_1(s,t)$ of S_f in $\tilde{B}$ to B and locate an appropriate (u_0, v_0) such that $\tilde{\Phi}(u_0, v_0)$ can be refined to the desired surface point. Since the local shape of S_f within a cube B is approximated by $\Phi_1(s,t)$, it is clear that the location of $\mathbf{x}_1$ closely influences the overall performance of the approximation. Ideally, we expect that $\mathbf{x}_1$ is approximately at the *center* of all surface intersections on edges of B. The approach we take is to choose (u_0, v_0) so that (u_0, v_0) is the barycenter of $(u_1, v_1), \ldots, (u_k, v_k)$ where $\tilde{\Phi}_1(u_1, v_1), \ldots, \tilde{\Phi}_1(u_k, v_k)$ are intersections of $\tilde{\Phi}_1(s,t)$ with edges of B.

Recall that with each transversal face F of $\tilde{B}$ there is an associated record

$$[F, < (e_1, p_1), (\bar{e}_1, (s_1, t_1)) >, < (e_2, p_2), (\bar{e}_2, (s_2, t_2)) >]$$

where, for $i = 1, 2$, p_i is the intersection of S_f on edge e_i of F, p_i is refined from $\tilde{\Phi}_1(s_i, t_i)$, and $\bar{e}_i$ is the edge of $\tilde{B}$ with which $\tilde{\Phi}_1(s,t)$ intersects at $\tilde{\Phi}_1(s_i, t_i)$. When $\bar{e}_1$ or $\bar{e}_2$ is not on F, it is clear that $\tilde{\Phi}_1$ is no longer appropriate for the above computation. In this case, we first compute $s_0 = \frac{1}{2}(s_1 + s_2)$ and $t_0 = \frac{1}{2}(t_1 + t_2)$ and refine $\tilde{\Phi}_1(s_0, t_0)$ to a surface point p of S_f on face F and then on p we derive a new approximant $\tilde{\Phi}_1(s,t)$; see Figure 4.8(a). This computation is also applied to the case in which $(s_1, t_1) = (s_2, t_2)$; see also Figure 4.8(b). Moreover, for better approximation of $\tilde{\Phi}_1(s,t)$ within B, this computation can generally be applied when both $\bar{e}_1$ and $\bar{e}_2$ are edges of the face F. Hence, as an uniform approach for all cases, we compute such a new approximant $\tilde{\Phi}_1(s,t)$ for the transversal face F.

Nevertheless, Newton iteration may fail to refine the computed $\tilde{\Phi}(u_0, v_0)$ to a surface point $(\mathbf{x}_1, \mathbf{x}_2)$ or the iteration succeeds but $\mathbf{x}_1$ is outside B. When this happens, the following heuristic approach can be applied:

1. compute $s_0 = \frac{1}{2}(s_0 + u_0)$ and $t_0 = \frac{1}{2}(t_0 + v_0)$.

2. refine $\tilde{\Phi}_1(s_0, t_0)$ to a surface point on which a new approximant $\tilde{\Phi}_1(s,t)$ is derived.

3. compute (u_0, v_0).

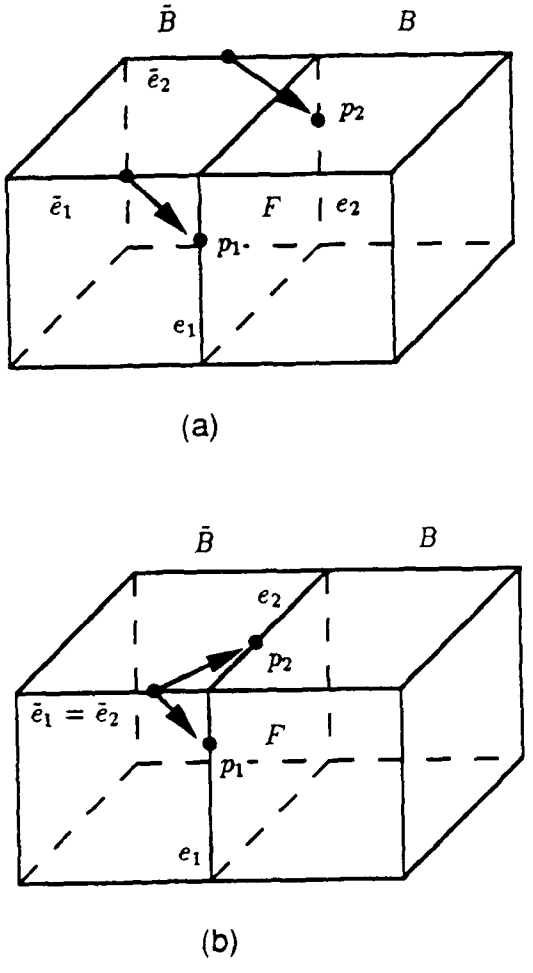


Figure 4.8 Special cases for stepping

4. refine $\tilde{\Phi}_1(u_0, v_0)$ to a surface point.

5. go to step 1 if the step 4 fails.

4.3.4 Newton Iteration

Newton iteration has been used in this chapter to refine an approximate zero $\mathbf{p}_0$ to a true zero $\mathbf{p}$ of the system of equations by generating a sequence of points $\mathbf{p}_1, \mathbf{p}_2, \ldots, \rightarrow \mathbf{p}$. Systems (4.2) and (4.3) are a 0-dimensional nonlinear systems $\mathbf{G}(\mathbf{x}) = \mathbf{0}$, where $\mathbf{G} : \mathbf{R}^m \rightarrow \mathbf{R}^m$ for some m. The Newton iteration for solving such systems is given by

$$D\mathbf{G}(\mathbf{p}_k)(\mathbf{p}_{k+1} - \mathbf{p}_k) = -\mathbf{G}(\mathbf{p}_k)$$

This is a linear system of equations for $\mathbf{p}_{k+1}$, and if $D\mathbf{G}(\mathbf{p}_k)$ is nonsingular, it can be solved by general linear solvers.

When refining an approximate to a surface point of $\mathbf{F}(\mathbf{x}) = \mathbf{0}$ by Newton iteration, we solve the following underdetermined linear system

$$\nabla f_i(\mathbf{p}_k) \cdot \Delta_k = -f_i(\mathbf{p}_k), \qquad i = 1, 2, \ldots, n-2 \tag{4.4}$$

where $\Delta_k = \mathbf{p}_{k+1} - \mathbf{p}_k$ and $D\mathbf{F}(\mathbf{p}_k)$ is of dimension $(n-2) \times n$. Equation (4.4) is the same as equation (3.12). When $D\mathbf{F}(\mathbf{p}_k)$ has rank $n-2$, the general solution is

$$\Delta_k = \alpha_1 \mathbf{t}_1 + \alpha_2 \mathbf{t}_2 + \beta_1 \nabla f_1(\mathbf{p}_k) + \cdots + \beta_{n-2} \nabla f_{n-2}(\mathbf{p}_k) \tag{4.5}$$

where $\mathbf{t}_1$ and $\mathbf{t}_2$ form a unit vector base of the tangent space of $\mathbf{F}(\mathbf{x}) = \mathbf{0}$ at $\mathbf{p}_k$.

If it is wished to make the computation more numerically stable, one may use singular value decomposition as we have done in Section 3.4. The matrix $(D\mathbf{F}(\mathbf{p}_k))^T$ is factored as $(D\mathbf{F}(\mathbf{p}_k))^T = U\Sigma V^T$, where $U = [\mathbf{u}_1, \ldots, \mathbf{u}_n] \in \mathbf{R}^{n \times n}$ and $V = [\mathbf{v}_1, \ldots, \mathbf{v}_{n-2}] \in \mathbf{R}^{(n-2) \times (n-2)}$ are orthogonal matrices, and $\Sigma \in \mathbf{R}^{n \times (n-2)}$ is a diagonal matrix. Directly substituting the factorization to system (4.4) we obtain

$$U\Sigma^T U^T \Delta_k = -\mathbf{F}(\mathbf{p}_k)$$

whose solution can be generally written as

$$\Delta_k = \gamma_1 \mathbf{u}_1 + \gamma_2 \mathbf{u}_2 + \cdots + \gamma_n \mathbf{u}_n$$

where $\gamma_1, \ldots, \gamma_{n-2}$ are uniquely determined by $\gamma_i = (-\mathbf{v}_i^T \mathbf{F}(\mathbf{p}_k))/\Sigma_{ii}$ and γ_{n-1} and γ_n are arbitrary. For Δ_k, we assign $\gamma_{n-1} = \gamma_n = 0$ so that Δ_k is in the normal space of S_F at $\mathbf{p}_k$ since $\mathbf{u}_1, \ldots, \mathbf{u}_{n-2}$ span the same space as the gradients $\nabla f_1 \mathbf{p}_k), \ldots, \nabla f_{n-2}(\mathbf{p}_k)$. Alternatively, we can compute the QR-factorization of $(D\mathbf{F}(\mathbf{p}_k))^T$ as

$$(D\mathbf{F}(\mathbf{p}_k))^T = \mathbf{Q} \begin{bmatrix} \mathbf{R} \\ \mathbf{0} \end{bmatrix}$$

where $\mathbf{Q}$ is a $n \times n$ orthogonal matrix and $\mathbf{R}$ is a $(n-2) \times (n-2)$ nonsingular, upper triangular matrix. To compute Δ_k, we

1. solve $\mathbf{R}^T \mathbf{y} = -\mathbf{F}(\mathbf{p}_k)$ for $\mathbf{y} \in \mathbf{R}^{n-2}$.

2. solve $\mathbf{Q}^T \Delta_k = (\mathbf{y}, \mathbf{0})^T$, i.e., $\Delta_k = \mathbf{Q}(\mathbf{y}, \mathbf{0})^T$.

Notice that the last 2 columns of $\mathbf{Q}$ forms an orthogonal basis of the tangent space of S_F at $\mathbf{p}_k$. Thus Δ_k so computed is a linear combination of the gradients.

4.3.5 Some Error Analyses

A problem of practical and theoretical interest is to estimate the error bound between the projected surface and its local parametric approximant within a cube or on cube's edges. The computation of this estimate would depend on error analyses of both the projection and the approximation. It is not clear now how to approach the above problem. Instead, we consider the estimation of the error bound between the local parameterization of the projected surface and the local parametric approximant within a cube, under the assumption that the neighborhood of convergence of the local parameterization covers the parametric points inside the cube.

Recall that the local parametric approximant $\Phi_1(s,t)$ of S_f consists of the initial terms of the Taylor series of the local parameterization up to degree 2. Let $\Psi_1(s,t)$ denote the local parameterization of S_f. Consider the coordinate functions $\phi_i(s,t)$ and $\psi_i(s,t)$ of $\Phi_1(s,t)$ and $\Psi_1(s,t)$ respectively. From the differential calculus,

$$\psi_i(s,t) = \phi_i(s,t) + R_i^{(3)}(\tau_i, s, t)$$

for some $0 \leq \tau_i \leq 1$, where $R_i^{(j)}(\tau_i, s, t) = \frac{1}{j!}((s,t) \cdot (\frac{\partial}{\partial s}, \frac{\partial}{\partial t}))^j \psi_i(\tau_i(s,t))$. Thus

$$|\psi_i(s,t) - \phi_i(s,t)| = |R_i^{(3)}(\tau_i, s, t)| \leq \max_{0 \leq \tau_i \leq 1} |R_i^{(3)}(\tau_i, s, t)|$$

However, $\psi_i(s,t)$ is unknown and hence the above bound cannot be computed. Consider one exact representation of the error

$$\psi_i(s,t) - \phi_i(s,t) = T_i^{(3)}(s,t) + R_i^{(4)}(\tau_i, s, t)$$

for $0 \leq \tau_i \leq 1$, where $T_i^{(j)}(s,t) = \frac{1}{j!}((s,t) \cdot (\frac{\partial}{\partial s}, \frac{\partial}{\partial t}))^j \psi_i(s,t)$. With $h = \|(s,t)\|$, $T_i^{(3)}(s,t)$ is an $O(h^4)$ accurate approximation of the error $\psi_i(s,t) - \phi_i(s,t)$. Thus the vector $(T_1^{(3)}(s,t), T_2^{(3)}(s,t), T_3^{(3)}(s,t))^T$ estimates $\Psi_1(s,t) - \Phi_1(s,t)$, componentwise, to within $O(h^4)$ accuracy, and hence

$$\|\Psi_1(s,t) - \Phi_1(s,t)\| = \|(T_1^{(3)}(s,t), T_2^{(3)}(s,t), T_3^{(3)}(s,t))^T\| + O(h^4)$$

Notice that the coefficients of $T_i^{(3)}(s,t)$ can be computed using equation (3.12).

4.3.6 Adaptive Subdivision

The cube size δ is an input to the PLA algorithm. To determine an appropriate δ for a given problem *a-priori* is nontrivial. In practice, we would expect a large δ initially, and perform adaptive subdivision whenever necessary. There are cases in which a cube must be subdivided in order to continue the computation. The cases are as follows.

1. When a parametric approximant penetrates a face without intersecting any edge of the face as shown in Figure 4.2.

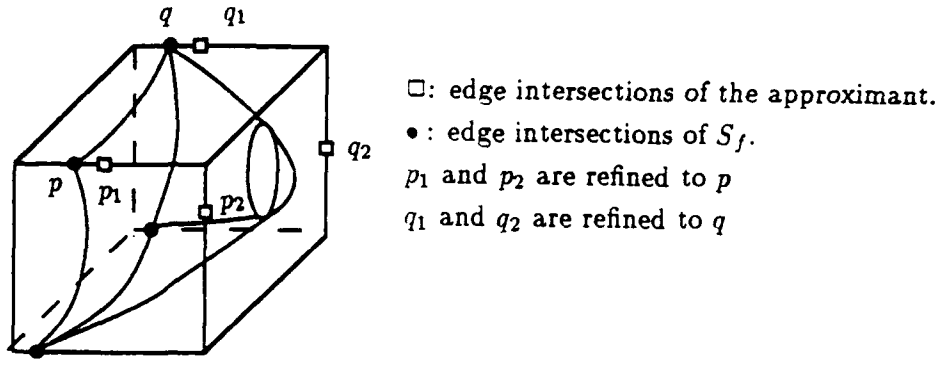


Figure 4.9 An invalid polygon

2. When Newton iteration fails to refine the intersections of the parametric approximant with edges of the cube to surface points on edges.

3. When an approximate point is refined to an unexpected surface point and hence an invalid polygon is produced. e.g., see Figure 4.9.

Like PLA methods that use vertex evaluation, some portions of the surface might be truncated if we do not subdivide. Two typical examples are

1. The projected surface S_f penetrates a face in a closed curve without intersecting the boundary edges; see Figure 4.2(a). In this case, both neighboring cubes sharing that face should be subdivided. This is difficult to detect in general.

2. S_f intersects only one edge of a face; i.e., S_f intersects an edge of a face at two points and has no intersections with others; e.g., see Figure 4.2(b). Thus all four neighboring cubes along that edge are subdivided for a reliable polygonization.

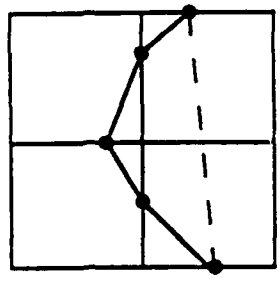


Figure 4.10 A crack on the shared face

Subdivision is performed by calling Algorithm 4.1 recursively, with the parent cube as the domain of interest and with the subcube containing the regular surface point as the starting cube. Because of size differences, cracks may occur along the transversal faces of the parent cube. See also Figure 4.10. In order to close such cracks, a data structure is needed that records topological adjacencies of cubes.

We modify the data structures of Algorithm 4.1 by associating with each parent cube a list of transversal subcubes, and pointers to neighboring transversal cubes at the same level. Thus for each cube B we have

$$< B, \Phi(s,t), face(B), (pt_1, pt_2, pt_3, pt_4, pt_5, pt_6), pt_p, L_B > \qquad (4.6)$$

where pt_p is a pointer to the parent cube, L_B, a list of elements in the form (4.6), is the list of transversal subcubes of B, and pt_i is a pointer to the adjacent cube sharing face F_i with B. The list L_B is assigned to the parent cube on the return from subdivision. The pointers pt_i to neighboring transversal cubes are assigned when such adjacencies are confirmed in step 8 of Algorithm 4.1. The $face(B)$ is derived from L_B if B is subdivided.

4.3.7 Polygonization and Local Refinement

The PLA of S_f is a polygonal representation given as a list of polygons derived from those cubes that intersect S_f. A list of transversal cubes lying in a given domain D is produced by Algorithm 4.1. For each transversal cube, say B, there is an associated list of transversal faces and surface intersections from which the polygon approximating S_f within B is formed.

When adaptive subdivision is incorporated into Algorithm 4.1, a transversal cube B might be recursively subdivided and hence the polygonization processing is confined to terminal transversal subcubes. As mentioned in Section 4.3.6, the polygonization scheme must close cracks along the shared face of two transversal cubes.

Recall that we associated with a face F of a cube C a record of intersection information in the following form

$$[F, < (e_1, p_1), (\bar{e}_1, (s_1, t_1)) >, < (e_2, p_2), (\bar{e}_2, (s_2, t_2)) >]$$

When tracing the transversal faces of a cube C in order to form a polygon over C, we check on each transversal face F to see if the adjacent cube $\tilde{C}$ has been subdivided. If $\tilde{C}$ is subdivided, possibly more than once, we trace the surface-edge intersections on face F of $\tilde{C}$ starting from p_1 on e_1. If the faces of a cube are numbered consistently, this computation can be achieved efficiently by processing the faces of the same number on the subcubes of $L_{\tilde{C}}$ that share F with C. Note that when C is a subcube of B while $\tilde{C}$ is not, $\tilde{C}$ can be located by following the parent pointers and the pointers to the neighboring cubes.

Once a polygon is formed, it is desirable to refine locally the polygon according to some criterion provided by the user, e.g., the maximum deviation of vertex normals from the normal of $p = \Phi_1(0,0)$. Let the polygon P be the list of vertices $[p_1, p_2, \ldots, p_k]$, for some k, and let N_i be the unit normal of S_f at p_i, $i = 1, \ldots, k$. Also let N be the unit normal at p. As shown in [17], the maximum deviation of

vertex normals from the normal of the p can be estimated by

$$\max_{1 \le i \le k} (N_i \cdot N)$$

When the maximum deviation exceeds some tolerance, polygon P is replaced by triangles $[p, p_i, p_{i+1}]$ for $i = 1, \ldots, k-1$, and each of the triangles is refined accordingly. Note that on the refinement of triangles, the midpoint of the triangle is refined to a surface point by Newton iteration.

Surface normal computations are also needed when shading the surface to produce an image. For computing the normals of the projected surface S_f directly from $\mathbf{F}(\mathbf{x}) = \mathbf{0}$, see Section 3.1.1. Notice that the coordinates of the polygon vertices in $\mathbf{R}^n$ are stored in order to do the refinement and compute surface normals.

5. CONCLUSIONS AND FUTURE WORK

5.1 Local Implicit Approximations of Curves and Surfaces

We have presented a method for computing a local implicit approximation of parametric curves and surfaces. The method works for both polynomially and rationally parameterized curves and surfaces, and achieves an order of contact that can be prescribed. In the case of nonsingular curve points, the approximant must be irreducible, but in the surface case additional safeguards have been incorporated into the algorithm to ensure irreducibility. The method also yields meaningful results for many types of singularity. The algorithm is capable of determining the exact implicit form without extraneous factors when the approximant is formulated with the exact degree of the implicit form.

The method provides a middle ground between two major approaches for evaluating the intersection curve of two parametric surfaces, that is, subdivision and substitution methods. It is well known that subdivision methods are robust and can locate all intersection branches, but at the expense of creating a large volume of data, while the substitution method provides an exact representation of the intersection with the help of often expensive implicitization techniques. In the context of subdivision methods, our implicit approximations have the potential of reducing the number of generated surface approximants since we are not restricted to only planar approximants. In the context of substitution methods, the approximations avoid the high cost of implicitization. In both cases a number of practical issues remain open for exploration, including the trade-off between the degree of the approximant and the accuracy with which the curve or surface has been approximated. In particular, a comparative evaluation of our method is

desirable that contrasts its performance with other surface intersection methods, such as the one based on subdivision.

5.2 Local Approximations of 2-Surfaces

We have demonstrated that certain surfaces, including offsets, blends and Voronoi surfaces, can be formulated as the projection, into 3-space, of 2-surfaces $\mathbf{F}(\mathbf{x}) = \mathbf{0}$ in $\mathbf{R}^n$. For such surfaces, we have proposed several computation schemes that (1) describe the local geometry, including computations of normal vectors, tangent vectors, and normal curvatures, of the projected surface, and (2) derive degree two local implicit, local explicit and local parametric approximations of the projected surface. We believe that these methods will be of practical interest in rendering 2-surfaces in high-dimensional space and computing surface/surface intersection.

In computing the degree two implicit approximation of the projected surface, an 8×10 linear system is obtained, which leaves us two degrees of freedom. Problems of practical and theoretical interest include describing the set of approximants parameterized by the two degrees of freedom and deriving criteria from which to determine a member of the family of approximants to select.

Several other problems are of interest, for example, computing the Gaussian and mean curvatures of the projected surface at a point directly from $\mathbf{F}(\mathbf{x}) = \mathbf{0}$ without variable elimination, and computing parametric approximations of the projected surface at singular surface points.

5.3 Piecewise Approximations of 2-Surfaces

The ability to derive a piecewise linear approximation (PLA) of a 2-surface defined implicitly in $\mathbf{R}^n$ should be essential in the interactive design environment since the PLA allows one to take advantage of hardware capabilities and reduces the cost of expensive ray tracing in the rendering process.

We have presented an algorithm that computes the PLA of a 2-surface defined by a system of nonlinear equations in n variables, where $n > 3$, but whose natural projection to 3-space is the surface of interest. This is an algorithm that deals with a 2-surface in $\mathbf{R}^n$, but performs all major computations in 3-space. Hence its performance on computing the PLA of complex surfaces, including offsets, blends, and Voronoi surfaces, is much more efficient than methods that work in n-space.

A number of issues await future research. Our algorithm takes as input a cube of a prescribed size containing a given point on the (projected) surface. One might ask how one should determine the cube size. In the process of the algorithm, a degree two parametric approximant is derived to approximate the projected surface within the cube and its intersections with the cube's edges serve as initial points to be refined to the intersections of the projected surface with edges of the cube. Moreover, cube subdivision can be easily applied when necessary and due to the availability of the parametric approximant the local refinement within a polygon can be efficiently performed. Thus, we would expect a large cube size initially, followed by adaptive subdivision. An ideal cube size would balance local geometry and global geometry such that only a small number of cube subdivisions are needed. Such trade-offs remain an important problem to be explored in greater depth.

Locating a seed point on the projected surface is in general difficult especially for blends and Voronoi surfaces. Space decomposition is useful for surfaces in $\mathbf{R}^3$, but seems expensive for 2-surfaces in $\mathbf{R}^n$ where n is large. It remains an urgent challenge for PLA algorithm based on continuation methods.

An efficient and reliable way to estimate stepping into an adjacent transversal cube is crucial to the performance of our algorithm. Although we have given a heuristic approach that is based on the local parametric approximant, a more quantitative method would be desirable that accounts for the local geometry of the projected surface.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] S. S. Abhyankar and C. Bajaj. Automatic parameterization of rational curves and surfaces I: Conics and conicoids. *Computer Aided Design*, 19(1):11–14, 1987.

[2] S. S. Abhyankar and C. Bajaj. Automatic parameterization of rational curves and surfaces II: Cubics and cubicoids. *Computer Aided Design*, 19(9):499–502, 1987.

[3] S. S. Abhyankar and C. Bajaj. Automatic parameterization of rational curves and surfaces III: Algebraic plane curves. *Computer Aided Geometric Design*, 5:309–321, 1988.

[4] S. S. Abhyankar and C. Bajaj. Automatic parameterization of rational curves and surfaces IV: Algebraic space curves. *ACM Transactions on Graphics*, 8(4):325–334, 1989.

[5] E. L. Allgower and K. Georg. Estimates for piecewise linear approximations of implicitly defined manifolds. *Applied Math. Letter*, 2(2):111–115, 1989.

[6] E. L. Allgower and K. Georg. *Numerical Continuation Methods, An Introduction*. Springer-Verlag, 1990. Springer Series in Computational Mathematics Vol. 13.

[7] E. L. Allgower and S. Gnutzmann. An algorithm for piecewise linear approximation of implicitly defined two-dimensional surfaces. *SIAM Journal on Numerical Analysis*, 24(2):452–469, 1987.

[8] E. L. Allgower and S. Gnutzmann. Polygonal meshes for implicitly defined surfaces, November 1989.

[9] E. L. Allgower and P. H. Schmidt. An algorithm for piecewise-linear approximation of an implicitly defined manifold. *SIAM Journal on Numerical Analysis*, 22(2):322–346, 1985.

[10] P. R. Arner. *Another Look at Surface/Surface Intersection*. PhD thesis, Department of Mathematics, University of Utah, 1987.

[11] C. L. Bajaj. Local parameterization, implicitization and inversion of real algebraic curves. Technical Report CSD-TR-863, CER-89-9, Computer Science Department, Purdue University, 1989.

[12] C. L. Bajaj. Rational hypersurface display. Technical Report CSD-TR-940, CER-90-2, Computer Science Department, Purdue University, 1990.

[13] C. L. Bajaj, T. Garrity, and J. Warren. On the applications of multi-equational resultants. Technical Report CSD-TR-826, Computer Science Department, Purdue University, November 1988.

[14] C. L. Bajaj, C. M. Hoffmann, R. E. Lynch, and J. E. H. Hopcroft. Tracing surface intersections. *Computer Aided Geometric Design*, 5:285–307, December 1988.

[15] C. L. Bajaj and I. Ihm. Hermite interpolation of rational space curves using real algebraic surfaces. In *Proceedings of the fifth Annual Symposium on Computational Geometry*, pages 94–103, Saarbrucken, West Geomany, June 1989. ACM.

[16] R. E. Barnhill, G. Farin, and B. R. Piper. Surface/surface intersection. *Computer Aided Geometric Design*, 4:3–16, 1987.

[17] J. Bloomenthal. Polygonization of implicit surfaces. *Computer Aided Geometric Design*, 5:341–355, 1988.

[18] B. Buchberger. Gröbner basis: An algorithmic method in polynomial ideal theory. In N. K. Bose, editor, *Multidimensional Systems Theory*, pages 184–232. D. Reidel Publishing Co., 1985.

[19] B. Buchberger, G. Collins, and B. Kutzler. Algebraic methods for geometric reasoning. *Annl. Reviews in Computer Science*, 3:85–120, 1988.

[20] J. Canny. Generalized characteristic polynomials. In *Proceedings of the 1988 Intl. Sym. on Symbolic and Algebraic Computation, ISAAC-88*, 1988. Rome, Italy.

[21] V. Chandru, D. Dutta, and C. M. Hoffmann. Variable radius blending with cyclides. In K. Preiss, J. Turner, and M. Wozny, editors, *Geometric Modeling for Product Engineering*, pages 39–58. North Holland, 1990.

[22] E.-W. Chionh. *Base Points, Resultants, and the Implicit Representation of Rational Surfaces.* PhD thesis, Department of Computer science, University of Waterloo, Canada, 1990.

[23] J. H. Chuang and C. M. Hoffmann. On local implicit approximation and its applications. *ACM Transactions on Graphics*, 8(4), October 1989.

[24] J. H. Chuang and C. M. Hoffmann. Curvature computations on surfaces in n-space, 1990. In Preparation.

[25] D. Dutta and C. M. Hoffmann. A geometric investigation of the skeleton of CSG objects. Technical Report CSD-TR-955, Computer Science Department, Purdue University, 1990.

[26] Gerald Farin. *Curves and Surfaces for Computer Aided Geometric Design, A Practical Guide.* Academic Press, Inc., 2 edition, 1988.

[27] R. T. Farouki. The characterization of parametric surface sections. *Computer Vision, Graphics, and Image Processing*, 1986.

[28] T. Garrity and J. Warren. On computing the intersection of a pair of algebraic surfaces. *Computer Aided Geometric Design*, 6:137–153, 1989.

[29] G. Golub and C. Van Loan. *Matric Computations.* John Hopkins University Press, 1977.

[30] C. M. Hoffmann. A dimensionality paradigm for surface interrogations. Technical Report CSD-TR-837, Computer Science Department, Purdue University, December 1988. to appear in CAGD.

[31] C. M. Hoffmann. *Geometric and Solid Modeling, An Introduction.* Morgan Kaufmann, San Francisco, 1989.

[32] C. M. Hoffmann. Algebraic and numerical techniques for offsets and blends. In W. Dahmen, M. Gasca, and C. Micchelli, editors, *Computations of Curves and Surfaces.* Kluwer Academic Publishers, 1990.

[33] C. M. Hoffmann. Conversion methods between parametric and implicit curves and surfaces. Technical Report CSD-TR-975, Computer Science Department, Purdue University, April 1990.

[34] C. M. Hoffmann and J. Hopcroft. Automatic surface generation in computer aided design. *The Visual Computer*, 1(2):92–100, 1985.

[35] C. M. Hoffmann and J. Hopcroft. Quadratic blending surfaces. *Computer Aided Design*, 18(6):301–306, 1986.

[36] C. M. Hoffmann and J. Hopcroft. The potential method for blending surfaces and corners. In G. Farin, editor, *Geometric Modeling: Algorithms and New Trends*, pages 347–365. SIAM Publications, 1987.

[37] C. M. Hoffmann and J. Hopcroft. The geometry of projective blending surfaces. *Artificial Intelligence*, 37:357–376, 1988.

[38] E. G. Houghton, R. F. Emnett, J. D. Factor, and C. L. Sabharwal. Implementation of a divide-and-conquer method for intersection of parametric surfaces. *Computer Aided Geometric Design*, 2, 1985.

[39] W. Kahan. A survey of error analysis. In *Proceedings of the International Federation for Information Processing Congress*, volume 2, pages 1214–1239. North-Holland, 1971.

[40] P. A. Koparkar and S. P. Mudur. Generation of continuous smooth curves resulting from operations on parametric surface patches. *Computer Aided Design*, 18(4), 1986.

[41] J. M. Lane and R. F. Riesenfeld. A theoretical development for the computer generation and display of piecewise polynomial surfaces. *IEEE Trans. Pattern Anal. and Mach. intelligence*, 2(1), 1980.

[42] D. Lasser. Intersection of parametric surfaces in the bernstein-bézier representation. *Computer Aided Design*, 18(4), 1986.

[43] D. Lazard. Gröbner bases, gaussian elimination and resolution of systems of algebraic equations. In EUROCAL '83, *Lecture Notes in Computer Science 162*, pages 146–156. Springer-Verlag, 1983.

[44] J. Z. Levin. Mathematical models for determining the intersection of quadric surfaces. *Computer Graphics and Image Processing*, 11:73–87, 1979.

[45] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *Computer Graphics*, 21(4):163–169, July 1987.

[46] F. S. Macaulay. Some formulae in elimination. *Proc. London Math. Soc.*, 35:3–27, 1902.

[47] F. S. Macauley. *The Algebraic Theory of Modular Systems*. Cambridge University Press, 1916.

[48] Y. de Montaudouin, W. Tiller, and H. Vold. Applications of power series in computational geometry. *Computer Aided Design*, 18(10):514–524, December 1986.

[49] Netto. *Algebra*. Midland Press, Ann Arbor, Michigan, 1896. Vol 1, p. 169, and Vol 2, p. 79.

[50] B. O'Neill. *Elementary Differential Geometry*. Academic Press, 1966.

[51] N. M. Patrikalakis, 1987. Oral Communication.

[52] J. Pegna and F.E. Wolter. A simple practical criterion to guarantee second order smoothness of blend surfaces, 1989.

[53] Q. S. Peng. An algorithm for finding the intersection lines between two b-spline surfaces. *Computer Aided Design*, 16(4), 1984.

[54] P. V. Prakash and N. M. Patrikalakis. Algebraic and rational polynomial surface intersections, 1988. Submitted to Computer Vision, Graphics and Image Processing.

[55] A. A. Requicha and H. B. Voelcker. Solid modeling: Current status and research directions. *IEEE Computer Graphics & Applications*, 3:25–37, October 1983.

[56] W. C. Rheinboldt. On a moving-frame algorithm and the triangulation of equilibrium manifolds. In T. Küpper, R. Seydel, and H. Troger, editors, *Bifurcation: Analysis, Algorithms, Applications*, pages 256–267. Brikhäuser Verlag Basel, 1987. International Series of Numerical Mathematics, Vol. 79.

[57] W. C. Rheinboldt. On the computation of multi-deimensional solution manifolds of parametrized equations. *Numerische Mathematik*, 53:165–181, 1988.

[58] G. Salmon. *Modern Higher Algebra*. Hodges, Smith, and Co., Dublin, 1866.

[59] R. F. Sarraga. Algebraic methods for intersections of quadric surfaces in GMSOLID. *Computer Vision, Graphics, and Image Processing*, 22, 1983.

[60] T. W. Sederberg. Degenerate parametric curves. *Computer Aided Geometric Design*, 1:301–307, 1984.

[61] T. W. Sederberg. *Implicit and Parametric Curves and Surfaces for Computer Aided Geometric Design*. PhD thesis, Purdue University, 1984.

[62] T. W. Sederberg. Improperly parametrized rational curves. *Computer Aided Geometric Design*, 3, 1986.

[63] T. W. Sederberg, D. Anderson, and R. Goldman. Implicit representation of parametric curves and surfaces. *Computer Vision, Graphics, and Image Processing*, 28:72–84, 1985.

[64] T. W. Sederberg and J. Snively. Parameterization of cubic algebraic surfaces. In R. Martin, editor, *Mathematics of Surfaces II*, pages 299–321. Oxford University Press, 1987.

[65] A. Tarski. *A Decision Method for Elementary Algebra and Geometry*. Berkeley: Univ. Calif. Press, 2 edition, 1951.

[66] J. A. Thorpe. *Elementary Topics in Differential Geometry*. Springer-Verlag, 1979.

[67] W. N. Waggenspack, Jr. *Parametric Curve Approximations for Surface Intersection.* PhD thesis, Purdue University, 1987.

[68] W. N. Waggenspack, Jr. and D. C. Anderson. Piecewise parametric approximations for algebraic curves. *Computer Aided Geometric Design*, 6:33 - 53, 1989.

[69] Wen-Tsun Wu. On a projection theorem of quasi-varieties in elimination theory, 1989.

VITA

VITA

Jung-Hong Chuang was born on March 8, 1956, in Tainan, Taiwan, Republic of China. He received the B.S. degree in Applied Mathematics from National Chiao Tung University, Taiwan, in 1978, the M.S. degree in Statistics from University of Pittsburgh, Pittsburgh, PA, in 1983, and the M.S. degree in Computer Science from Purdue University, West Lafayette, IN, in 1986.

His research interests include solid modeling, computer aided geometric design, computer graphics, and the applications of algebraic and differential geometry to solid modeling. While at Purdue he was supported as a teaching assistant and as a research assistant under the supervision of Professor Christoph M. Hoffmann.